دانشگاه صنعتی اصفهان

دانشکده برق و کامپیوتر

تمرین اول درس امنیت شبکه

مقدمه ای بر رمزنگاری

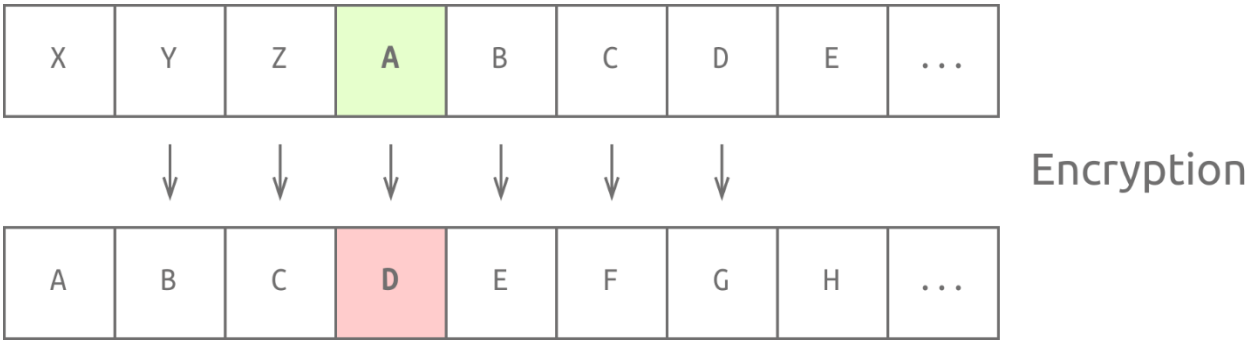
بخش ۱: مقدمه

هدف این بخش آشنا کردن کاربران با مفاهیم پایه رمزنگاری مانند:

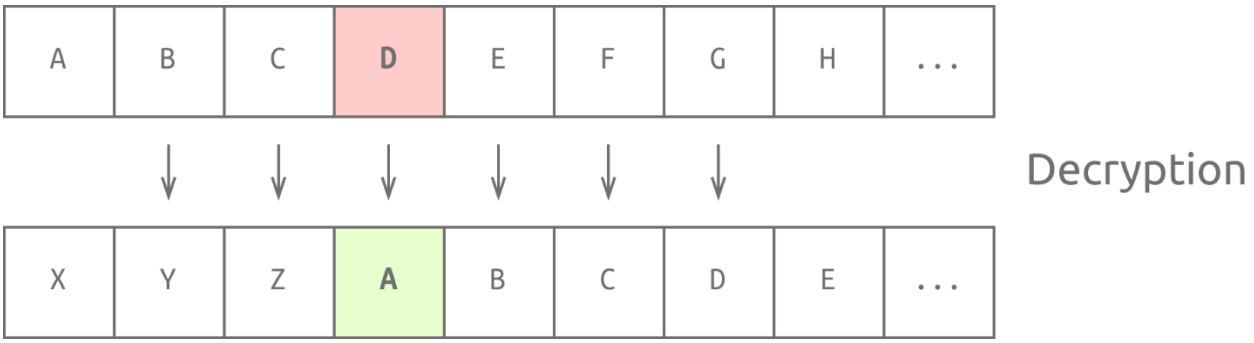
- رمزنگاری متقارن، مانند AES
- رمزنگاری نامتقارن، مانند RSA
- تبادل کلید دیفی- هلمن
- هش گذاری
- PKI

فرض کنید می‌خواهید پیامی را ارسال کنید که هیچ‌کس به جز دریافت‌کننده مورد نظر نتواند آن را درک کند. چگونه این کار را انجام می‌دهید؟

یکی از ساده‌ترین رمزها، رمز سزار است که بیش از ۲۰۰۰ سال پیش استفاده می‌شد. رمز سزار حروف را به تعداد مشخصی به سمت چپ یا راست جابجا می‌کند. فرض کنید که جابجایی به اندازه ۳ به سمت راست برای رمزگذاری استفاده شود.



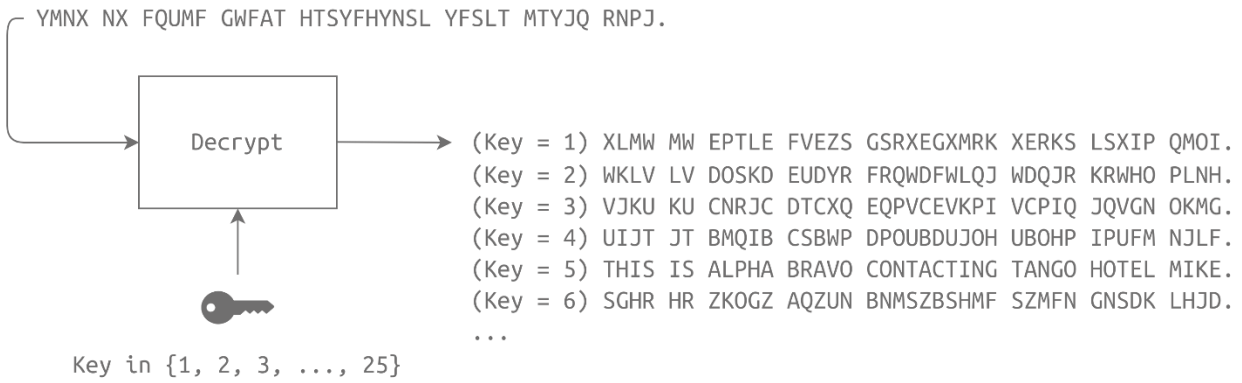
دریافت‌کننده باید بداند که متن به اندازه ۳ به سمت راست جابجا شده تا بتواند پیام اصلی را بازیابی کند.



با استفاده از همان کلید برای رمزگذاری "HELLO WORLD"، نتیجه "KHOOR ZRUOG" به دست می‌آید.

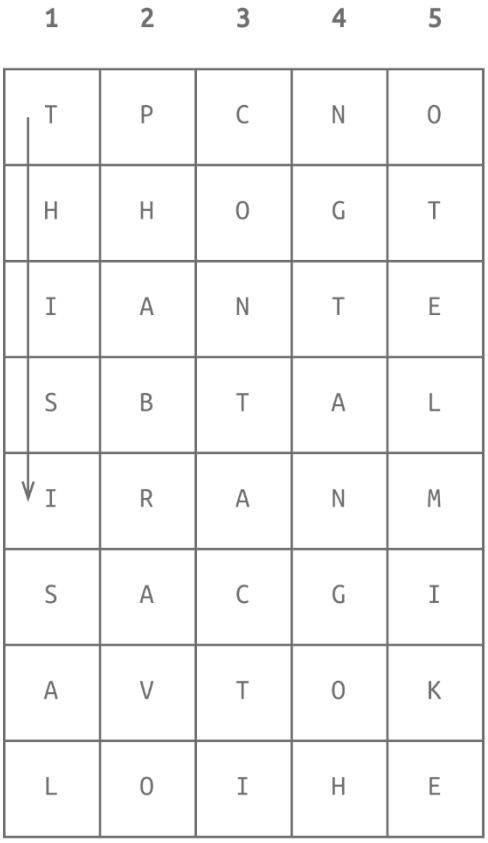
رمز سزاری که در بالا توصیف کردیم می‌تواند کلیدی بین ۱ تا ۲۵ داشته باشد. با کلید ۱، هر حرف به یک موقعیت جابجا می‌شود، جایی که A به B و Z به A تبدیل می‌شود. با کلید ۲۵، هر حرف به ۲۵ موقعیت جابجا می‌شود، جایی که A به Z و B به A تبدیل می‌شود. کلید ۰ به معنی عدم تغییر است و به علاوه، کلید ۲۶ نیز هیچ تغییری ایجاد نمی‌کند چرا که به چرخش کامل منجر می‌شود. بنابراین، نتیجه می‌گیریم که رمز سزار دارای فضای کلید ۲۵ است؛ یعنی ۲۵ کلید مختلف وجود دارد که کاربر می‌تواند انتخاب کند.

فرض کنید پیامی را که با استفاده از رمز سزار رمزگذاری شده است، شنود کرده‌اید: “YMNX NX FQUMF GWFAT HTSYFHYNSL YFSLT MTYJQ RNPJ”. از ما خواسته شده است که بدون اطلاع از کلید آن را رمزگشایی کنیم. می‌توانیم این کار را با استفاده از روش جستجوی تمام کلیدها انجام دهیم و ببینیم کدام کلید منطقی‌تر است. در شکل زیر متوجه می‌شویم که کلید ۵ منطقی‌ترین نتیجه را به ما می‌دهد: “THIS IS ALPHA BRAVO CONTACTING TANGO HOTEL MIKE”.



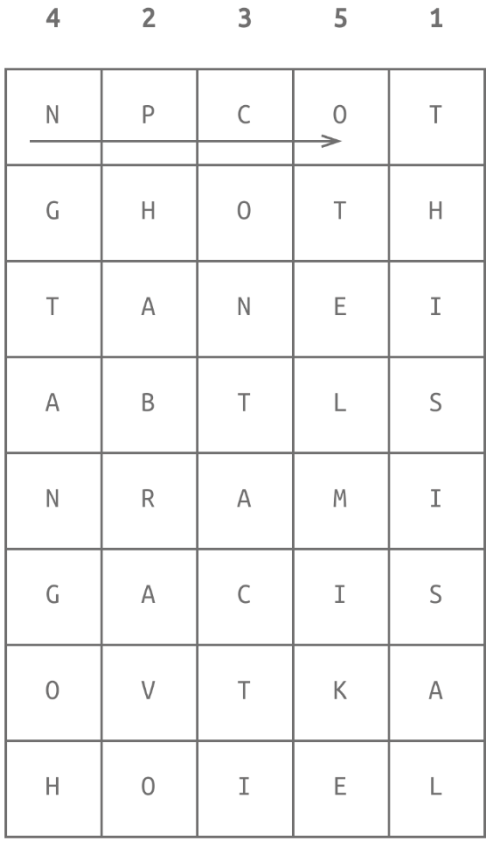
رمز سزار به عنوان یک رمز جانشینی در نظر گرفته می‌شود چرا که هر حرف در الفبا با حرف دیگری جایگزین می‌شود.

نوع دیگری از رمز، رمز جابجایی است که پیام را با تغییر ترتیب حروف رمزگذاری می‌کند. بیایید یک رمز جابجایی ساده را در شکل زیر در نظر بگیریم. ما با پیام " THIS IS ALPHA BRAVO CONTACTING TANGO HOTEL MIKE" و کلید ۴۲۳۵۱ شروع می‌کنیم. پس از اینکه حروف پیام خود را با پر کردن یک ستون بعد از دیگری نوشتیم، ستون‌ها را بر اساس کلید مرتب می‌کنیم و سپس ردیف‌ها را می‌خوانیم. به عبارت دیگر، ما با ستون‌ها می‌نویسیم و با ردیف‌ها می‌خوانیم. همچنین توجه داشته باشید که ما در این مثال تمام فاصله‌ها را در متن اصلی نادیده گرفته‌ایم. متن رمز شده "NPCOTGHOTH..." به صورت ردیف به ردیف خوانده می‌شود. به عبارت دیگر، یک رمز جابجایی به سادگی ترتیب حروف را جابجا می‌کند، بر خلاف رمز جانشینی که حروف را بدون تغییر ترتیب آن‌ها جایگزین می‌کند.



Plaintext

THIS IS ALPHA BRAVO
CONTACTING TANGO HOTEL MIKE



Ciphertext

NPCOTGHOTHTANEIABTLS
NRAMIGACISOVTKAHOIEL

این بخش، رمزهای جانشینی و جابجایی ساده را معرفی کرد و آن‌ها را به پیام‌های متشکل از کاراکترهای الفبایی اعمال کرد. برای اینکه یک الگوریتم رمزگذاری ایمن در نظر گرفته شود، باید امکان بازیابی پیام اصلی (متن اصلی) غیرممکن باشد. (به زبان ریاضی، ما به یک مسأله سخت نیاز داریم، یعنی مسأله‌ای که نتوان آن را در زمان چندجمله‌ای حل کرد. مسأله‌ای که می‌توان آن را در زمان چندجمله‌ای حل کرد، مسأله‌ای است که حتی برای ورودی‌های بزرگ، قابل حل است، هرچند ممکن است برای اتمام آن زمان زیادی نیاز باشد.)

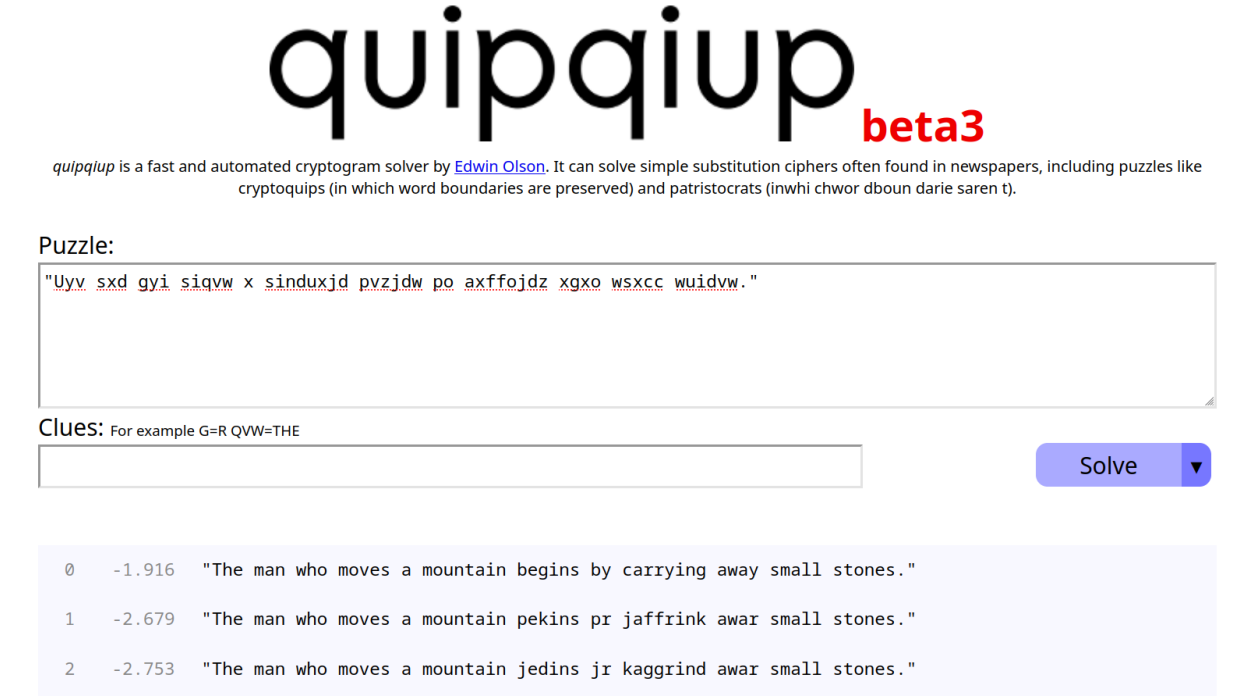
اگر پیام رمزگذاری شده در یک هفته قابل شکستن باشد، رمزگذاری استفاده شده ناامن در نظر گرفته می‌شود. با این حال، اگر پیام رمزگذاری شده در یک میلیون سال قابل شکستن باشد، رمزگذاری از نظر عملی امن در نظر گرفته می‌شود.

به رمز جانشینی تک‌الفبایی فکر کنید، جایی که هر حرف به یک حرف جدید نگاشت می‌شود. برای مثال، در زبان انگلیسی، شما "a" را به یکی از ۲۶ حرف انگلیسی نگاشت می‌کنید، سپس "b" را به یکی از ۲۵ حرف باقی‌مانده نگاشت می‌کنید و به همین ترتیب.

برای مثال، ممکن است حروف الفبا "abcdefghijklmnopqrstuvwxyz" را به "xpatvrzyjhecsdikbfwunqgmol" نگاشت کنیم. به عبارت دیگر، "a" به "x"، "b" به "p" و غیره تبدیل می‌شود. دریافت‌کننده باید کلید "xpatvrzyjhecsdikbfwunqgmol" را بداند تا بتواند پیام‌های رمزگذاری شده را به درستی رمزگشایی کند.

این الگوریتم ممکن است بسیار ایمن به نظر برسد، به‌ویژه چون تلاش برای تمام کلیدهای ممکن غیرممکن است. با این حال، تکنیک‌های مختلفی می‌توانند برای شکستن متن رمز شده با استفاده از چنین الگوریتم رمزگذاری استفاده شوند. یکی از ضعف‌های این الگوریتم، فراوانی حروف است. در متون انگلیسی، متداول‌ترین حروف 'e' ، 't' و 'a' هستند که به ترتیب با فراوانی ۱۳٪، ۹.۱٪ و ۸.۲٪ ظاهر می‌شوند. به علاوه، در متون انگلیسی، متداول‌ترین حروف اول، 't'، 'a' و 'o' هستند که به ترتیب با فراوانی ۱۶٪، ۱۱.۷٪ و ۷.۶٪ ظاهر می‌شوند. با این حال، بیشتر کلمات پیام‌ها کلمات دیکشنری هستند و شما می‌توانید یک متن رمز شده با رمز جانشینی الفبایی را به راحتی بشکنید.

ما واقعاً نیازی به استفاده از کلید رمزگذاری برای رمزگشایی متن رمز شده دریافتی " Uyv sxd gyi siqvw x sinduxjd pvzjdw po axffojdz xgxo wsxcc wuidvw." نداریم. همان‌طور که در شکل زیر نشان داده شده است، با استفاده از سایتی مانند quipqiup، تنها چند لحظه طول می‌کشد تا کشف کنیم که متن اصلی " The man who moves a mountain begins by carrying away small stones." بوده است. این مثال به وضوح نشان می‌دهد که این الگوریتم شکسته شده و نباید برای ارتباطات محرمانه استفاده شود.



به سؤالات زیر پاسخ دهید

شما پیام رمزگذاری شده زیر را دریافت کرده‌اید:

“Xjnvw lc sluxjmw jsqm wjpmcqbg jg wqcxqmnvw; xjzjmmjd lc wjpm sluxjmw jsqm bqccqm zqy.” Zlwvzjxj Zpcvcol

این پیام را رمزگشایی کرده و متن اصلی را بنویسید.

بخش ۲: رمزنگاری متقارن

بیایید برخی از اصطلاحات را مرور کنیم:

الگوریتم رمزنگاری یا رمز: این الگوریتم فرآیندهای رمزگذاری و رمزگشایی را تعریف می‌کند.

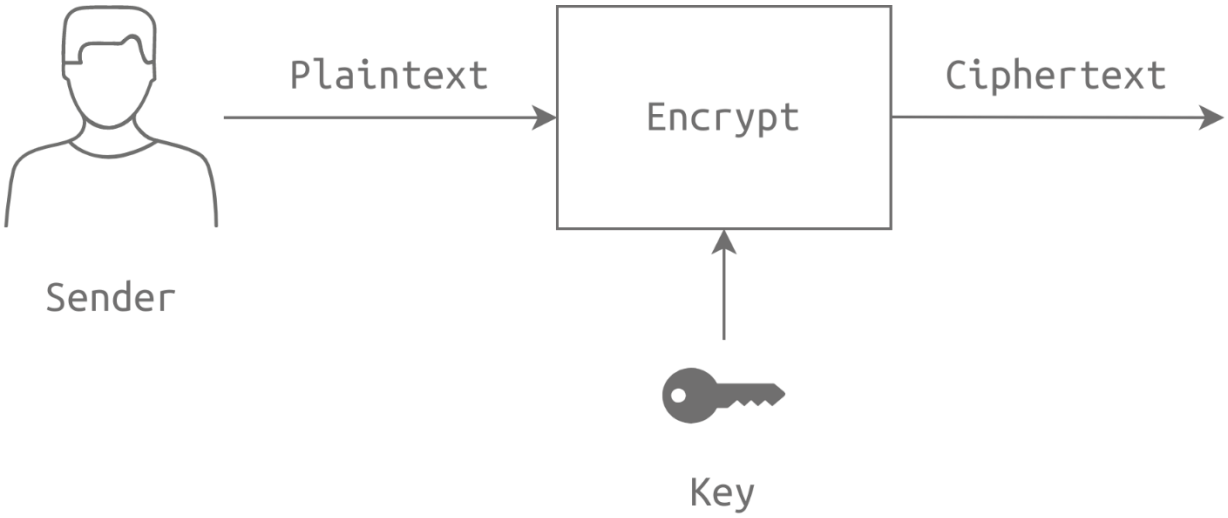
کلید: الگوریتم رمزنگاری برای تبدیل متن اصلی به متن رمز شده و بالعکس به یک کلید نیاز دارد.

متن اصلی: پیام اصلی است که می‌خواهیم رمزگذاری کنیم.

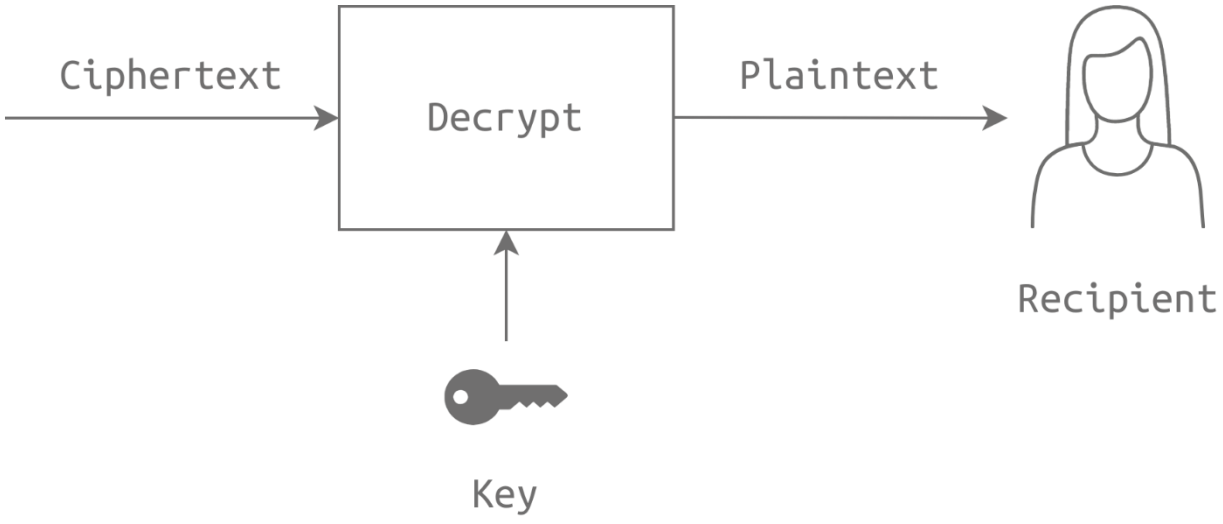
متن رمز شده: پیام به فرم رمزگذاری شده است.

الگوریتم رمزنگاری متقارن از همان کلید برای رمزگذاری و رمزگشایی استفاده می‌کند. بنابراین، طرفین در حال ارتباط باید قبل از تبادل هرگونه پیام، بر روی یک کلید مخفی توافق کنند.

در شکل زیر، فرستنده فرآیند رمزگذاری را با متن اصلی و کلید ارائه می‌دهد تا متن رمز شده را به دست آورد. متن رمز شده معمولاً از طریق یک کانال ارتباطی ارسال می‌شود.



در سمت دیگر، گیرنده فرآیند رمزگشایی را با همان کلید مورد استفاده فرستنده ارائه می‌دهد تا متن اصلی را از متن رمز شده دریافتی بازیابی کند. بدون دانش از کلید، گیرنده قادر به بازیابی متن اصلی نخواهد بود.

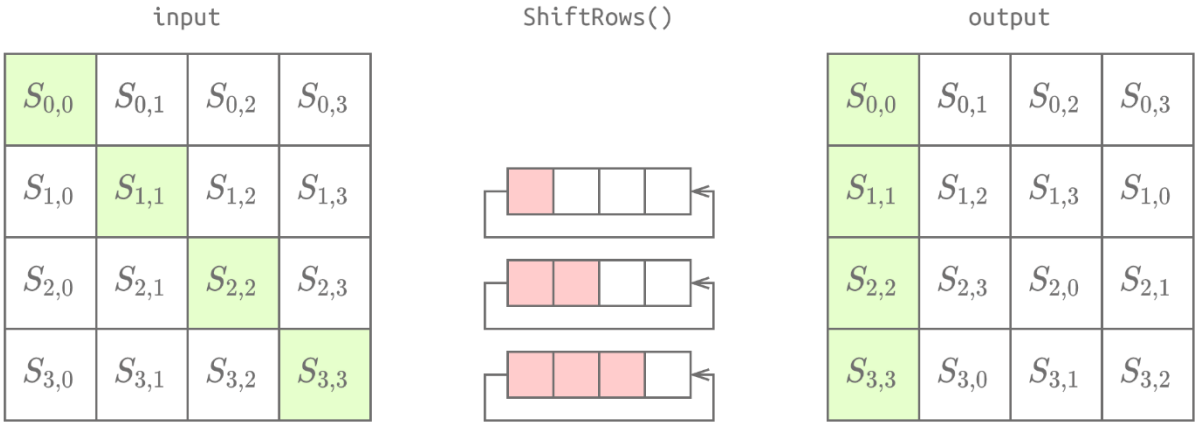


موسسه ملی استاندارد و فناوری (NIST) در سال ۱۹۷۷ استاندارد رمزگذاری داده‌ها (DES) را منتشر کرد. DES یک الگوریتم رمزنگاری متقارن است که از اندازه کلید ۵۶ بیت استفاده می‌کند. در سال ۱۹۹۷، چالشی برای شکستن

یک پیام رمز شده با DES حل شد و نشان داد که استفاده از جستجوی بی‌رحمانه برای یافتن کلید و شکستن یک پیام رمز شده با DES ممکن شده است. در سال ۱۹۹۸، یک کلید DES در ۵۶ ساعت شکسته شد. این موارد نشان داد که دیگر نمی‌توان DES را به عنوان یک الگوریتم امن در نظر گرفت.

NIST در سال ۲۰۰۱ استاندارد رمزگذاری پیشرفته (AES) را منتشر کرد. مانند DES، این یک الگوریتم رمزنگاری متقارن است؛ با این حال، از اندازه کلید ۱۲۸، ۱۹۲ یا ۲۵۶ بیت استفاده می‌کند و هنوز هم به عنوان یک الگوریتم امن و مورد استفاده در نظر گرفته می‌شود. AES چهار تبدیل زیر را چندین بار تکرار می‌کند:

۱. SubBytes(state): این تبدیل هر بایت را در یک جدول جانشینی خاص (S-box) جستجو کرده و با مقدار مربوطه جایگزین می‌کند. حالت ۱۶ بایت، یعنی ۱۲۸ بیت، در یک آرایه ۴ در ۴ ذخیره می‌شود.
۲. ShiftRows(state): ردیف دوم به یک مکان، ردیف سوم به دو مکان و ردیف چهارم به سه مکان جابجا می‌شود.
۳. MixColumns(state): هر ستون با یک ماتریس ثابت (آرایه ۴ در ۴) ضرب می‌شود.
۴. AddRoundKey(state): یک کلید دور به حالت با استفاده از عمل XOR اضافه می‌شود.



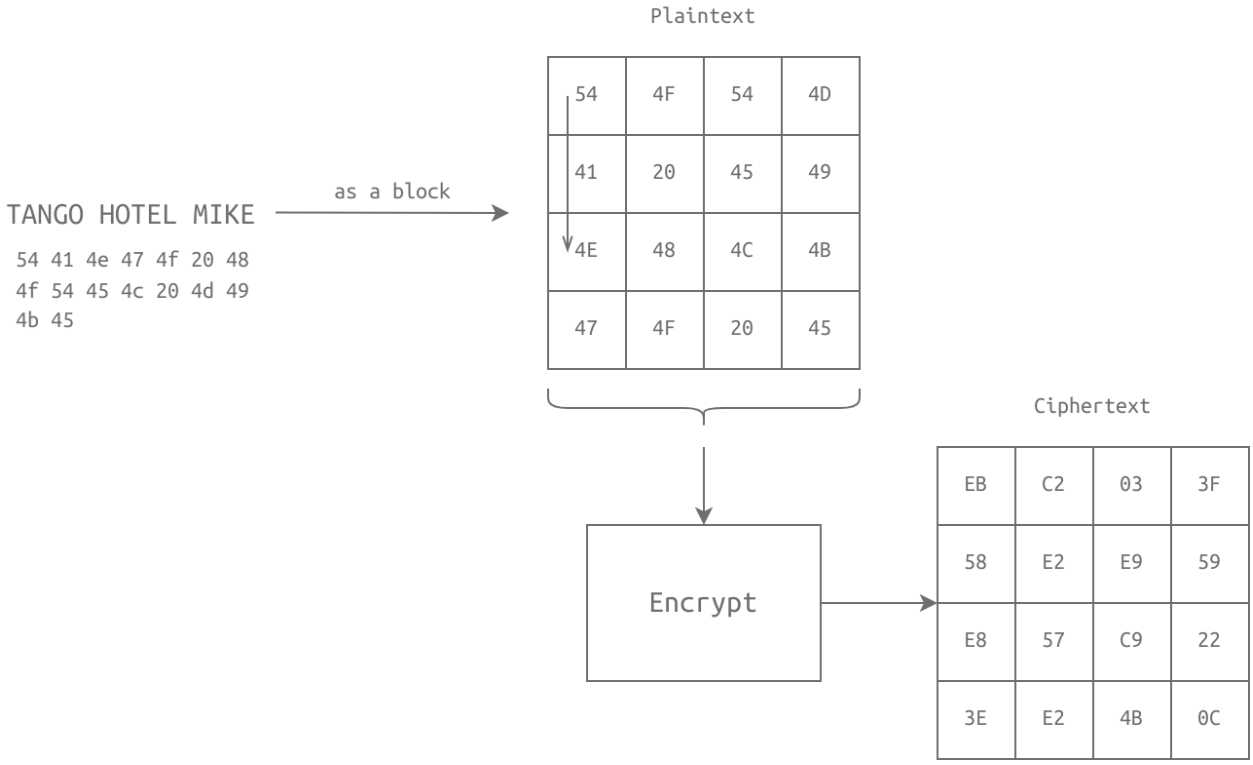
تعداد کل مراحل تبدیل به اندازه کلید بستگی دارد.

نگران نباشید اگر این موارد کمی رمزآلود به نظر برسند؛ هدف ما یادگیری جزئیات نحوه عملکرد AES یا پیاده‌سازی آن به عنوان یک کتابخانه برنامه‌نویسی نیست، بلکه هدف ما درک تفاوت در پیچیدگی بین الگوریتم‌های رمزنگاری قدیمی و مدرن است. اگر کنجکاو هستید که جزئیات بیشتری را بررسی کنید، می‌توانید مشخصات AES را بررسی کنید که شامل کدهای شبه و مثال‌ها در استاندارد منتشر شده آن، FIPS PUB 197 است.

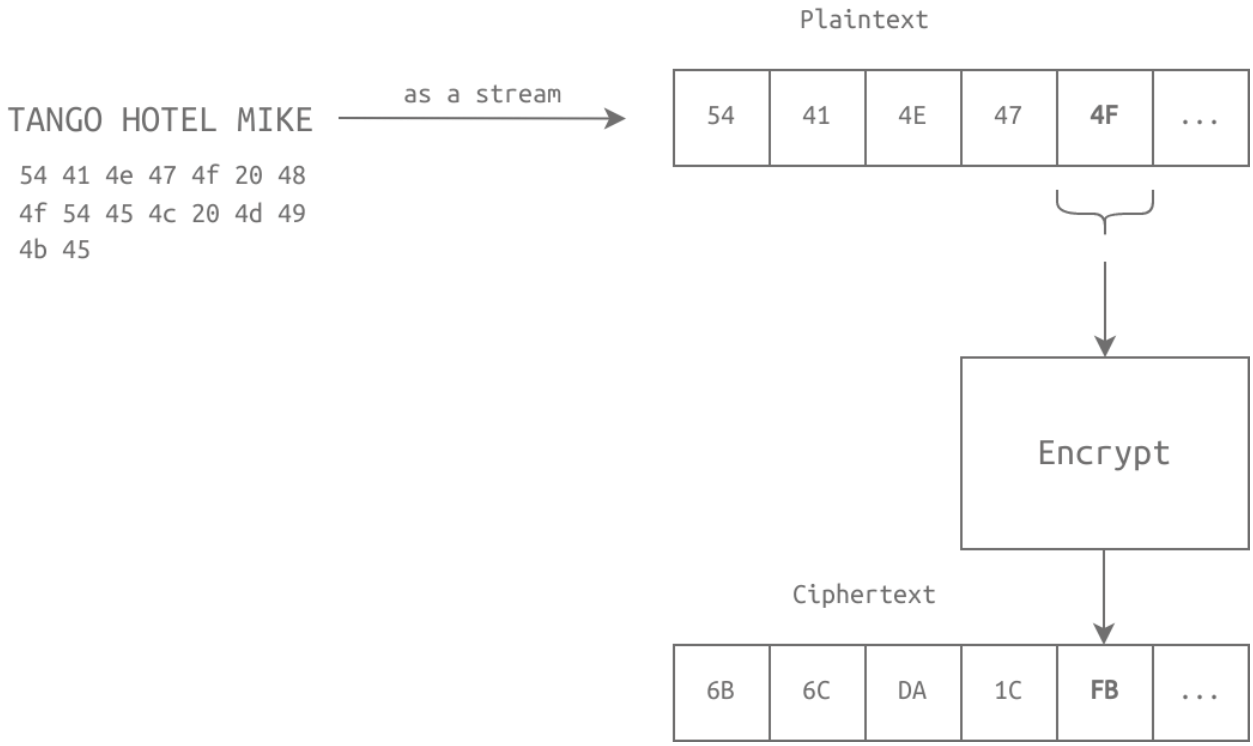
علاوه بر AES، الگوریتم‌های رمزنگاری متقارن دیگری نیز وجود دارند که به عنوان امن در نظر گرفته می‌شوند. در اینجا لیستی از الگوریتم‌های رمزنگاری متقارن که توسط GPG (GnuPG) 2.37.7 پشتیبانی می‌شود، آورده شده است:

الگوریتم رمزنگاری	توضیحات
AES، AES192 و AES256	AES با اندازه کلید ۱۲۸، ۱۹۲ و ۲۵۶ بیت
IDEA	الگوریتم رمزنگاری داده‌های بین‌المللی (IDEA)
3DES	DES سه‌گانه و بر اساس DES است. باید توجه داشته باشیم که DES۳ در سال ۲۰۲۳ منسوخ خواهد شد و در سال ۲۰۲۴ غیرمجاز خواهد بود.
CAST5	همچنین به عنوان CAST-128 شناخته می‌شود. برخی منابع می‌گویند که CAST نام نویسندگان آن، کارلایل آدامز و استافورد تاوارس است.
BLOWFISH	طراحی شده توسط بروس شنیئر
TWOFISH	طراحی شده توسط بروس شنیئر و مشتق شده از Blowfish
CAMELLIA128، CAMELLIA192 و CAMELLIA256	طراحی شده توسط میتسوبیشی الکتریک و NTT در ژاپن. نام آن از گل کاملیا ژاپنیکا گرفته شده است.

تمام الگوریتم‌های ذکر شده تا کنون، الگوریتم‌های رمزنگاری متقارن با بلوک رمز هستند. یک الگوریتم رمزگذاری بلوک ورودی (متن اصلی) را به بلوک‌ها تبدیل می‌کند و هر بلوک را رمزگذاری می‌کند. یک بلوک معمولاً ۱۲۸ بیت است. در شکل زیر، می‌خواهیم متن اصلی "TANGO HOTEL MIKE" را رمزگذاری کنیم که شامل ۱۶ کاراکتر است. اولین مرحله تبدیل آن به باینری است. اگر از ASCII استفاده کنیم، "T" به صورت x۵۴۰ در فرمت هگزادسیمال، "A" به صورت x۴۱۰ و غیره است. هر دو رقم هگزادسیمال به ۸ بیت تشکیل‌دهنده یک بایت تبدیل می‌شود. یک بلوک ۱۲۸ بیتی عملاً ۱۶ بایت است و در یک آرایه ۴ در ۴ نمایش داده می‌شود. بلوک ۱۲۸ بیتی به عنوان یک واحد به روش رمزگذاری داده می‌شود.



نوع دیگر الگوریتم رمزنگاری متقارن، رمزگذاری جریانی است که متن اصلی را بایت به بایت رمزگذاری می‌کند. فرض کنید می‌خواهیم پیام "TANGO HOTEL MIKE" را رمزگذاری کنیم؛ هر کاراکتر باید به نمایندگی باینری خود تبدیل شود. اگر از ASCII استفاده کنیم، "T" به صورت x54۰ در هگزادسیمال و "A" به صورت x41۰ و غیره است. روش رمزگذاری هر بار یک بایت را پردازش می‌کند. این در شکل زیر نشان داده شده است.



رمزنگاری متقارن بسیاری از مشکلات امنیتی که دراصول امنیتی بحث شده است را حل می‌کند. فرض کنید آلیس و باب ملاقات کرده و یک الگوریتم رمزگذاری را انتخاب کرده و بر روی یک کلید خاص توافق کرده‌اند. فرض می‌کنیم که الگوریتم انتخابی امن است و کلید مخفی به‌طور ایمن نگه‌داری می‌شود. بیایید نگاهی به آنچه می‌توانیم به دست آوریم بیندازیم:

- محرمانگی: اگر اوو پیام رمز شده را قطع کند، او نمی‌تواند متن اصلی را بازیابی کند. بنابراین، تمام پیام‌های مبادله شده بین آلیس و باب محرمانه هستند به شرطی که به صورت رمز شده ارسال شوند.
- تمامیت: هنگامی که باب یک پیام رمز شده را دریافت کرده و با موفقیت با استفاده از کلیدی که با آلیس توافق کرده است رمزگشایی می‌کند، باب می‌تواند مطمئن باشد که هیچ کس نمی‌تواند پیام را در طول کانال دستکاری کند. در استفاده از الگوریتم‌های رمزنگاری مدرن امن، هر گونه تغییر جزئی در متن رمز شده مانع از رمزگشایی موفق می‌شود یا به متنی بی‌معنی منجر می‌شود.
- اصالت: توانایی رمزگشایی متن رمز شده با استفاده از کلید مخفی همچنین اصالت پیام را ثابت می‌کند زیرا تنها آلیس و باب کلید مخفی را می‌دانند.

ما فقط در حال شروع هستیم و می‌دانیم چگونه محرمانگی را حفظ کنیم، تمامیت را بررسی کنیم و از اصالت پیام‌های مبادله شده اطمینان حاصل کنیم. رویکردهای عملی و کارآمدتری در وظایف بعدی ارائه خواهد شد. سوال در حال حاضر این است که آیا این مقیاس‌پذیر است.

با آلیس و باب، ما به یک کلید نیاز داشتیم. اگر آلیس، باب و چارلی داشته باشیم، به سه کلید نیاز داریم: یکی برای آلیس و باب، دیگری برای آلیس و چارلی و سومی برای باب و چارلی. با این حال، تعداد کلیدها به سرعت افزایش می‌یابد؛ ارتباط بین ۱۰۰ کاربر نیاز به تقریباً ۵۰۰۰ کلید مخفی مختلف دارد. (اگر کنجکاو هستید درباره ریاضیات پشت آن، این است: ۹۹ + ۹۸ + ۹۷ + ... + ۱ = ۴۹۵۰).

علاوه بر این، اگر یک سیستم به خطر بیفتد، آن‌ها باید کلیدهای جدیدی ایجاد کنند که با دیگر ۹۹ کاربر استفاده شود. مشکل دیگر پیدا کردن یک کانال امن برای تبادل کلیدها با تمام کاربران دیگر است. واضح است که این مسئله به سرعت از کنترل خارج می‌شود.

در بخش بعدی، ما به رمزنگاری نامتقارن خواهیم پرداخت. یکی از مشکلاتی که با رمزنگاری نامتقارن حل می‌شود این است که ۱۰۰ کاربر تنها به ۱۰۰ کلید برای ارتباط امن نیاز دارند. (همانطور که قبلاً توضیح داده شد، رمزنگاری متقارن تقریباً ۵۰۰۰ کلید برای ایمن‌سازی ارتباطات ۱۰۰ کاربر نیاز دارد.)

برنامه‌های زیادی برای رمزنگاری متقارن وجود دارد. ما بر روی دو مورد که به طور گسترده‌ای برای رمزنگاری نامتقارن نیز استفاده می‌شوند تمرکز خواهیم کرد:

- GNU Privacy Guard
- OpenSSL Project

GNU Privacy Guard

GNU Privacy Guard، که به GnuPG یا GPG نیز معروف است، استاندارد OpenPGP را پیاده‌سازی می‌کند.

می‌توانیم با استفاده از GnuPG (GPG) یک فایل را با استفاده از دستور زیر رمزگذاری کنیم:

```
gpg --symmetric --cipher-algo CIPHER message.txt
```

که در آن CIPHER نام الگوریتم رمزگذاری است. می‌توانید با استفاده از دستور gpg --version الگوریتم‌های رمزگذاری پشتیبانی‌شده را بررسی کنید. فایل رمزگذاری شده به صورت message.txt.gpg ذخیره می‌شود.

خروجی پیش‌فرض به فرمت باینری OpenPGP است؛ اما اگر ترجیح می‌دهید خروجی ASCII محافظت‌شده‌ای ایجاد کنید که در هر ویرایشگری باز شود، باید گزینه armor-- را اضافه کنید. برای مثال:

gpg --armor --symmetric --cipher-algo CIPHER message.txt.

می‌توانید با استفاده از دستور زیر رمزگشایی کنید:

gpg --output original_message.txt --decrypt message.gpg

پروژهOpenSSL

پروژه OpenSSL نرم‌افزار OpenSSL را توسعه می‌دهد.

می‌توانیم با استفاده از OpenSSL یک فایل را با استفاده از دستور زیر رمزگذاری کنیم:

openssl aes-256-cbc -e -in message.txt -out encrypted_message

می‌توانیم فایل حاصل را با استفاده از دستور زیر رمزگشایی کنیم:

openssl aes-256-cbc -d -in encrypted_message -out original_message.txt

برای افزایش امنیت رمزگذاری و مقاومت در برابر حملات، می‌توانیم از گزینه -pbkdf2 برای استفاده از تابع تولید کلید مبتنی بر رمز عبور ۲ (PBKDF2) استفاده کنیم؛ همچنین می‌توانیم تعداد تکرارها در رمز عبور برای استخراج کلید رمزگذاری را با استفاده از -iter NUMBER مشخص کنیم. برای تکرار ۱۰,۰۰۰ بار، دستور قبلی به شکل زیر تبدیل می‌شود:

openssl aes-256-cbc -pbkdf2 -iter 10000 -e -in message.txt -out encrypted_message

بنابراین، دستور رمزگشایی به این شکل می‌شود:

openssl aes-256-cbc -pbkdf2 -iter 10000 -d -in encrypted_message -out original_message.txt

فایل‌های مورد نیاز برای این بخش در فولدر task2 قرار دارند. فایل فشرده‌ای که به این بخش پیوست شده است، می‌تواند برای پاسخگویی به سوالات بخش‌های بعدی استفاده شود.

به سوالات زیر پاسخ دهید

فایل quote01 را که با کلید s!kR3T55 با AES256 رمزگذاری شده است، با gpg رمزگشایی کنید. متن رمزگشایی شده را به همراه دستورات در پاسخ نامه ارسال کنید.

فایل quote02 را که با کلید s!kR3T55 با AES256-CBC رمزگذاری شده است، با openssl رمزگشایی کنید. متن رمزگشایی شده را به همراه دستورات در پاسخ نامه ارسال کنید.

فایل quote03 را که با کلید s!kR3T55 با CAMELLIA256 رمزگذاری شده است، با gpg رمزگشایی کنید. متن رمزگشایی شده را به همراه دستورات در پاسخ‌نامه ارسال کنید.

بخش ۳: رمزنگاری نامتقارن

رمزنگاری متقارن نیاز به یافتن یک کانال امن برای تبادل کلیدها دارد. در اینجا، بیشتر به محرمانگی و یکپارچگی توجه داریم، یعنی کانالی که هیچ شخص سومی نتواند به آن گوش دهد یا داده‌ها را تغییر دهد.

رمزنگاری نامتقارن امکان تبادل پیام‌های رمزگذاری شده را بدون نیاز به کانال امن فراهم می‌کند؛ تنها نیاز به یک کانال مطمئن داریم که بیشتر بر یکپارچگی آن تمرکز داریم.

با استفاده از الگوریتم رمزنگاری نامتقارن، یک جفت کلید تولید می‌کنیم: یک کلید عمومی و یک کلید خصوصی. کلید عمومی با دیگران به اشتراک گذاشته می‌شود، در حالی که کلید خصوصی باید به طور امن نگه‌داری شود و هیچ‌کس نباید به آن دسترسی داشته باشد. همچنین، امکان استنتاج کلید خصوصی با دانستن کلید عمومی وجود ندارد.

چگونه این جفت کلید کار می‌کند؟

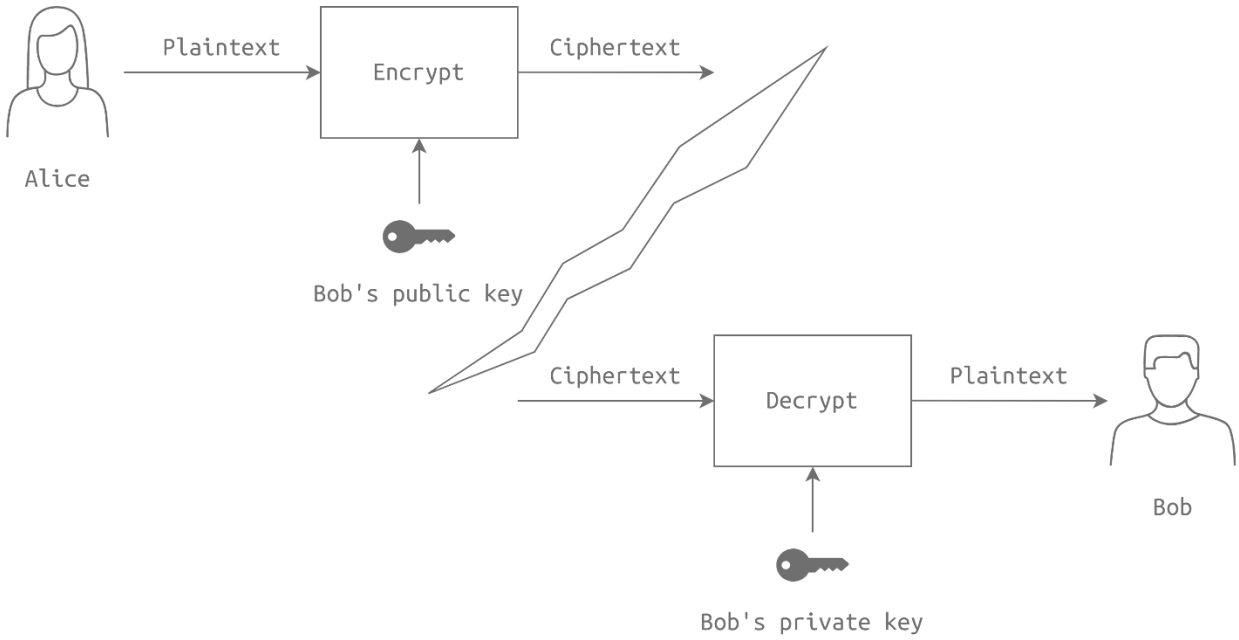
اگر یک پیام با یکی از کلیدها رمزگذاری شود، با کلید دیگر می‌توان آن را رمزگشایی کرد. به عبارت دیگر:

- اگر آلیس با کلید عمومی باب پیام را رمزگذاری کند، تنها با کلید خصوصی باب می‌توان آن را رمزگشایی کرد.
- برعکس، اگر باب پیام را با کلید خصوصی خود رمزگذاری کند، تنها با کلید عمومی‌اش می‌توان آن را رمزگشایی کرد.

محرمانگی

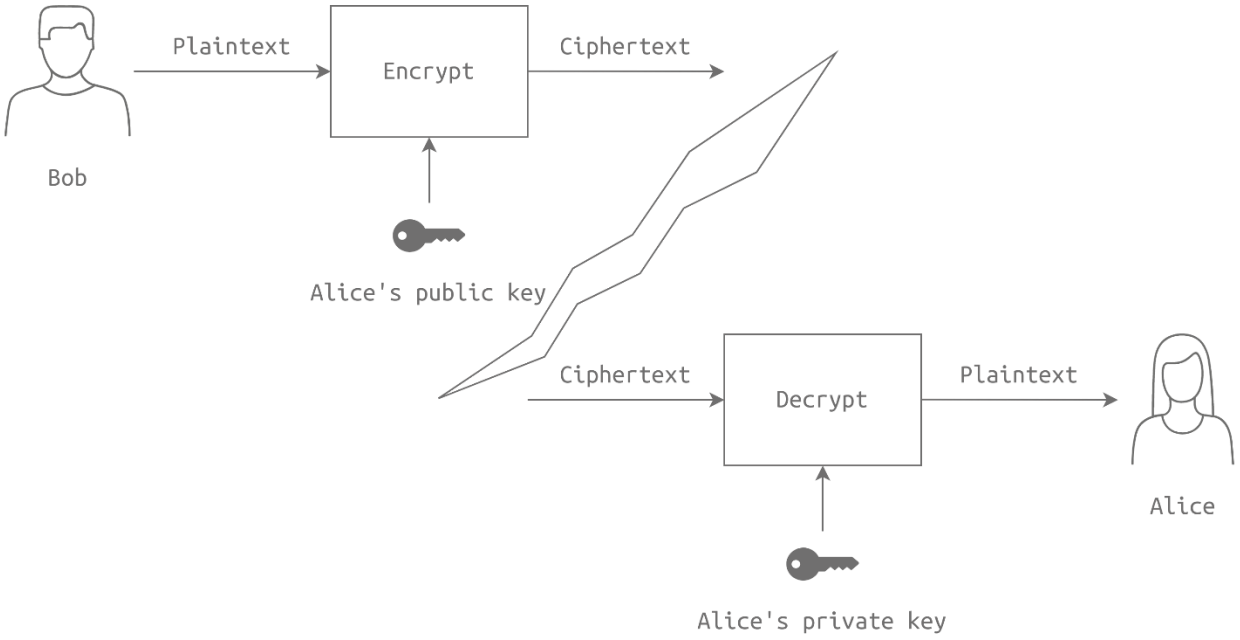
ما می‌توانیم از رمزنگاری نامتقارن برای دستیابی به محرمانگی با رمزگذاری پیام‌ها با استفاده از کلید عمومی گیرنده استفاده کنیم. در دو تصویر زیر، می‌توانیم ببینیم که:

آلیس می‌خواهد محرمانگی در ارتباط خود با باب را تضمین کند. او پیام را با استفاده از کلید عمومی باب رمزگذاری می‌کند و باب آن را با استفاده از کلید خصوصی خود رمزگشایی می‌کند. انتظار می‌رود که کلید عمومی باب در یک پایگاه داده عمومی یا وب‌سایت او منتشر شود.



هنگام استفاده از رمزنگاری نامتقارن، آلیس پیام‌ها را با استفاده از کلید عمومی باب قبل از ارسال به او رمزگذاری می‌کند. باب پیام‌ها را با استفاده از کلید خصوصی خود رمزگشایی می‌کند.

وقتی باب می‌خواهد به آلیس پاسخ دهد، او پیام‌های خود را با استفاده از کلید عمومی آلیس رمزگذاری می‌کند و آلیس می‌تواند آن‌ها را با استفاده از کلید خصوصی خود رمزگشایی کند.



به عبارت دیگر، ارتباط با آلیس و باب در حالی که محرمانگی پیام‌ها تضمین شده است، آسان می‌شود. تنها نیاز این است که همه طرف‌ها کلیدهای عمومی خود را برای فرستندگان علاقه‌مند در دسترس داشته باشند.

توجه: در عمل، الگوریتم‌های رمزنگاری متقارن عملیات سریع‌تری نسبت به رمزنگاری نامتقارن انجام می‌دهند؛ بنابراین، ما بعداً بررسی خواهیم کرد که چگونه می‌توانیم از بهترین‌های هر دو جهان بهره ببریم.

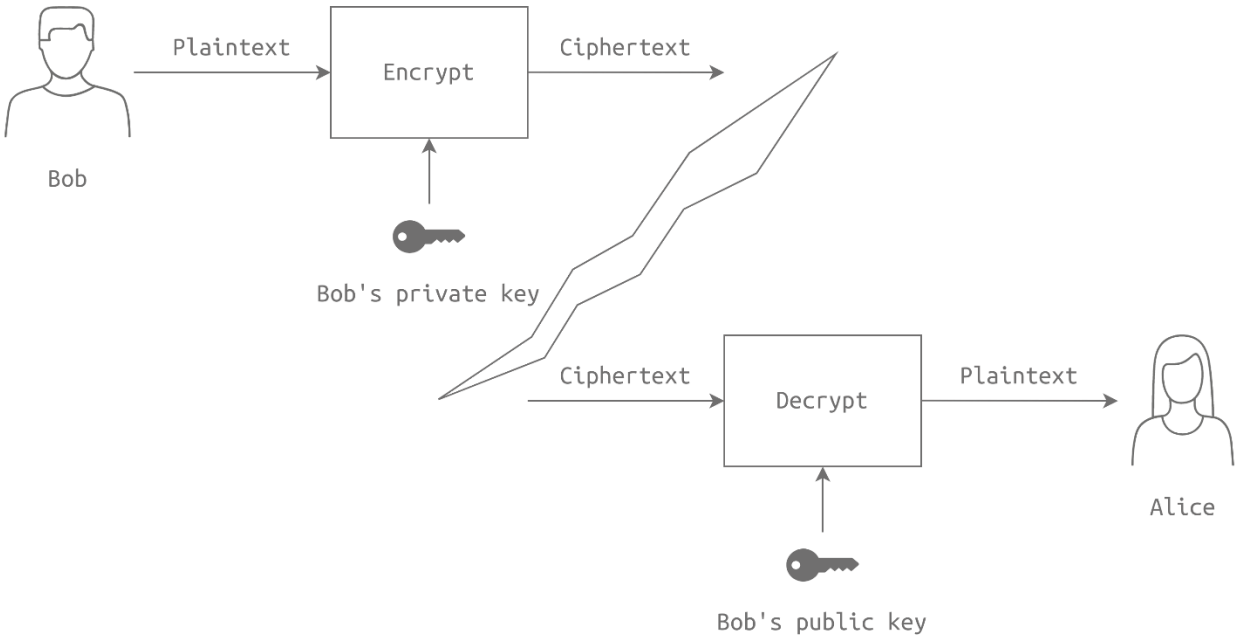
یکپارچگی، اعتبار و عدم انکار

فراتر از محرمانگی، رمزنگاری نامتقارن می‌تواند مسائل یکپارچگی، اعتبار و عدم انکار را حل کند. فرض کنید باب می‌خواهد بیانیه‌ای صادر کند و می‌خواهد همه بتوانند تأیید کنند که این بیانیه واقعاً از اوست. باب باید پیام را با استفاده از کلید خصوصی خود رمزگذاری کند؛ دریافت‌کنندگان می‌توانند آن را با استفاده از کلید عمومی باب رمزگشایی کنند. اگر پیام با کلید عمومی باب به‌درستی رمزگشایی شود، به این معنی است که پیام با کلید خصوصی باب رمزگذاری شده است.

رمزگشایی موفقیت‌آمیز با استفاده از کلید عمومی باب به چند نتیجه جالب منجر می‌شود:

- اول، پیام در طول مسیر (کانال ارتباطی) تغییر نکرده است؛ این یکپارچگی پیام را اثبات می‌کند.

- دوم، با توجه به اینکه هیچ‌کس به کلید خصوصی باب دسترسی ندارد، می‌توانیم مطمئن باشیم که این پیام واقعاً از باب آمده است؛ این اعتبار پیام را اثبات می‌کند.
- سرانجام، چون هیچ‌کس غیر از باب به کلید خصوصی او دسترسی ندارد، باب نمی‌تواند ارسال این پیام را انکار کند؛ این عدم انکار را برقرار می‌کند.



ما دیدیم که چگونه رمزنگاری نامتقارن می‌تواند به ایجاد محرمانگی، یکپارچگی، اعتبار و عدم انکار کمک کند. در سناریوهای واقعی، رمزنگاری نامتقارن می‌تواند نسبتاً کند باشد برای رمزگذاری فایل‌های بزرگ و مقادیر زیاد داده. در بخش دیگری، خواهیم دید که چگونه می‌توانیم از رمزنگاری نامتقارن به همراه رمزنگاری متقارن برای دستیابی به این اهداف امنیتی نسبتاً سریع‌تر استفاده کنیم.

RSA

RSA نام خود را از مخترعانش، Rivest، Shamir و Adleman گرفته است. نحوه کار آن به این صورت است:

۱. دو عدد اول تصادفی، p و q را انتخاب کنید. $N = p \times q$ را محاسبه کنید.
۲. دو عدد صحیح e و d را انتخاب کنید به‌گونه‌ای که $e \times d = 1 \bmod \phi(N)$، که در آن $\phi(N) = N - p - q + 1$. این مرحله به ما اجازه می‌دهد کلید عمومی (N,e) و کلید خصوصی (N,d) را تولید کنیم.
۳. فرستنده می‌تواند یک مقدار x را با محاسبه $y = x^e \bmod N$ رمزگذاری کند.

۴. گیرنده می‌تواند y را با محاسبه $x = y^d \bmod N$ رمزگشایی کند.

اگرچه معادلات ریاضی بالا ممکن است پیچیده به نظر برسند، شما نیازی به دانش ریاضی برای استفاده از RSA ندارید، زیرا این الگوریتم به‌راحتی از طریق برنامه‌ها و کتابخانه‌های برنامه‌نویسی در دسترس است.

امنیت RSA به سختی مسأله فاکتورگیری وابسته است. ضرب p در q آسان است؛ اما یافتن p و q با توجه به N زمان‌بر است. برای اینکه این الگوریتم امن باشد، p و q باید اعداد نسبتاً بزرگی باشند، به‌عنوان مثال، هرکدام ۱۰۲۴ بیت. نکته مهم این است که RSA به تولید عدد تصادفی امن وابسته است، مانند سایر الگوریتم‌های رمزنگاری نامتقارن. اگر یک مهاجم بتواند p و q را حدس بزند، کل سیستم غیر امن خواهد بود.

بیایید یک مثال عملی را بررسی کنیم.

۱. باب دو عدد اول انتخاب می‌کند: $p = 157$ و $q = 199$. او $N = 31243$ را محاسبه می‌کند.
۲. با $\phi(N) = N - p - q + 1 = 31243 - 157 - 199 + 1 = 30888$، باب $e = 163$ و $d = 379$ را انتخاب می‌کند که $e \times d = 163 \times 379 = 61777$ و ۶۱۷۷۷ $\bmod 30888 = 1$. کلید عمومی (۳۱۲۴۳,۱۶۳) و کلید خصوصی (۳۱۲۴۳,۳۷۹) هستند.
۳. فرض کنید که مقدار x برای رمزگذاری ۱۳ است، سپس آلیس محاسبه می‌کند و $y = 13^{163} \bmod 31243 = 16342$ را ارسال می‌کند.
۴. باب مقدار دریافتی را با محاسبه $x = 16341^{379} \bmod 31243 = 13$ رمزگشایی می‌کند.

این مثال برای درک بهتر ریاضیات پشت آن بود. برای مشاهده مقادیر واقعی p و q، بیایید یک جفت کلید واقعی با استفاده از ابزاری مانند openssl ایجاد کنیم.

```
openssl genrsa -out private-key.pem 2048
openssl rsa -in private-key.pem -pubout -out public-key.pem
writing RSA key
cat public-key.pem
```

```
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAymcAeYg1ohPQLHu7u911
UutN8bCP7r6czRX2zrQrpE1Yrm5mHERi1xweWEhTJ/0Q13FJcHLGtLbdQc0rGpOd
DnYJBuzrqXU2hC7E7d1qLsj63NPADq1OGYCGCWnm/HGM2WuVtDXqRitN4zeNKEWI
QmEctfucopZx5AVJ1vTn+qMv/0D6QU7Mm65MTSYg1SCRA0D0N9NLMj4rY1LOIr5q
5g3iunAE4tCROMcHf7fxWMuWdJTdtxTv7+4P5XGkWrWriO22JFHp9N22Fm96V9jH
7aASRkIZvQFmx+1d17btZDhsm2ezU07LBabv9efj0gIwz6P3mTJVm+wxaDH6jiXB
dwIDAQAB
-----END PUBLIC KEY-----
```

openssl rsa -in private-key.pem -text -noout

ما سه دستور را اجرا کردیم:

openssl genrsa -out private-key.pem 2048: با استفاده از openssl، ما از genrsa برای تولید یک کلید خصوصی RSA استفاده کردیم. با استفاده از -out، مشخص کردیم که کلید خصوصی تولیدی به‌عنوان private-key.pem ذخیره شود. ما ۲۰۴۸ را اضافه کردیم تا اندازه کلید ۲۰۴۸ بیت باشد.

openssl rsa -in private-key.pem -pubout -out public-key.pem: با استفاده از openssl، مشخص کردیم که از الگوریتم RSA با گزینه rsa استفاده می‌کنیم. مشخص کردیم که می‌خواهیم کلید عمومی را با استفاده از -pubout دریافت کنیم. در نهایت، کلید خصوصی را به‌عنوان ورودی با استفاده از -in private-key.pem تعیین کردیم و خروجی را با استفاده از -out public-key.pem ذخیره کردیم.

openssl rsa -in private-key.pem -text -noout: ما کنجکاو بودیم تا مقادیر واقعی RSA را ببینیم، بنابراین از -text -noout استفاده کردیم. مقادیر p، q، N، e و d به‌ترتیب prime1، prime2، modulus، publicExponent و privateExponent هستند.

اگر ما قبلاً کلید عمومی گیرنده را داشته باشیم، می‌توانیم آن را با دستور openssl pkeyutl -encrypt -in plaintext.txt -out ciphertext -inkey public-key.pem -pubin رمزگذاری کنیم.

گیرنده می‌تواند آن را با دستور openssl pkeyutl -decrypt -in ciphertext -inkey private-key.pem -out decrypted.txt رمزگشایی کند.

به سؤالات زیر پاسخ دهید:

از محتویات پوشه task3 برای حل سوالات زیر استفاده کنید.

باب فایل ciphertext_message را که از آلیس برایش ارسال شده، دریافت کرده است. کلید مورد نیاز را می‌توانید در همان پوشه پیدا کنید. متن اصلی چیست؟ (تصویر به همراه دستورات در پاسخ‌نامه ارسال گردد.)

به کلید خصوصی RSA باب نگاهی بیندازید. عدد p چه مقداری دارد؟

به کلید خصوصی RSA باب نگاهی بیندازید. عدد q چه مقداری دارد؟

بخش ۴: مبادله کلید دیفی-هلمن

آلیس و باب می‌توانند از طریق یک کانال ناامن ارتباط برقرار کنند. به این معنی که افرادی می‌توانند مکالمات رد و بدل شده در این کانال را بخوانند. آلیس و باب چگونه می‌توانند در چنین شرایطی بر سر یک کلید مخفی توافق کنند؟ یکی از راه‌ها استفاده از مبادله کلید دیفی-هلمن است.

دیفی-هلمن یک الگوریتم رمزنگاری نامتقارن است. این الگوریتم اجازه می‌دهد که یک راز از طریق یک کانال عمومی تبادل شود. ما از مبنای ریاضی صرف‌نظر می‌کنیم و یک مثال عددی ساده ارائه می‌دهیم. ما به دو عمل ریاضی نیاز داریم: توان و مدول. x^p، یعنی x به توان p، x را p بار در خودش ضرب می‌کند. همچنین، x mod m، یعنی x مدول m، باقی‌مانده تقسیم x بر m است.

۱. آلیس و باب بر روی q و g توافق می‌کنند. برای اینکه این کار کند، q باید یک عدد اول باشد و g عددی کمتر از q باشد که شرایط خاصی را برآورده کند. (در حساب مدولار، g یک تولیدکننده است.) در این مثال، ما $q = 29$ و $g = 3$ را در نظر می‌گیریم.
۲. آلیس یک عدد تصادفی a کمتر از q انتخاب می‌کند. او $A = (g^a) \bmod q$ را محاسبه می‌کند. عدد a باید به‌صورت مخفی نگه‌داشته شود؛ اما A به باب ارسال می‌شود. فرض کنیم آلیس عدد $a = 13$ را انتخاب کرده و $A = 3^{13}\%29 = 19$ را محاسبه کرده و به باب می‌فرستد.

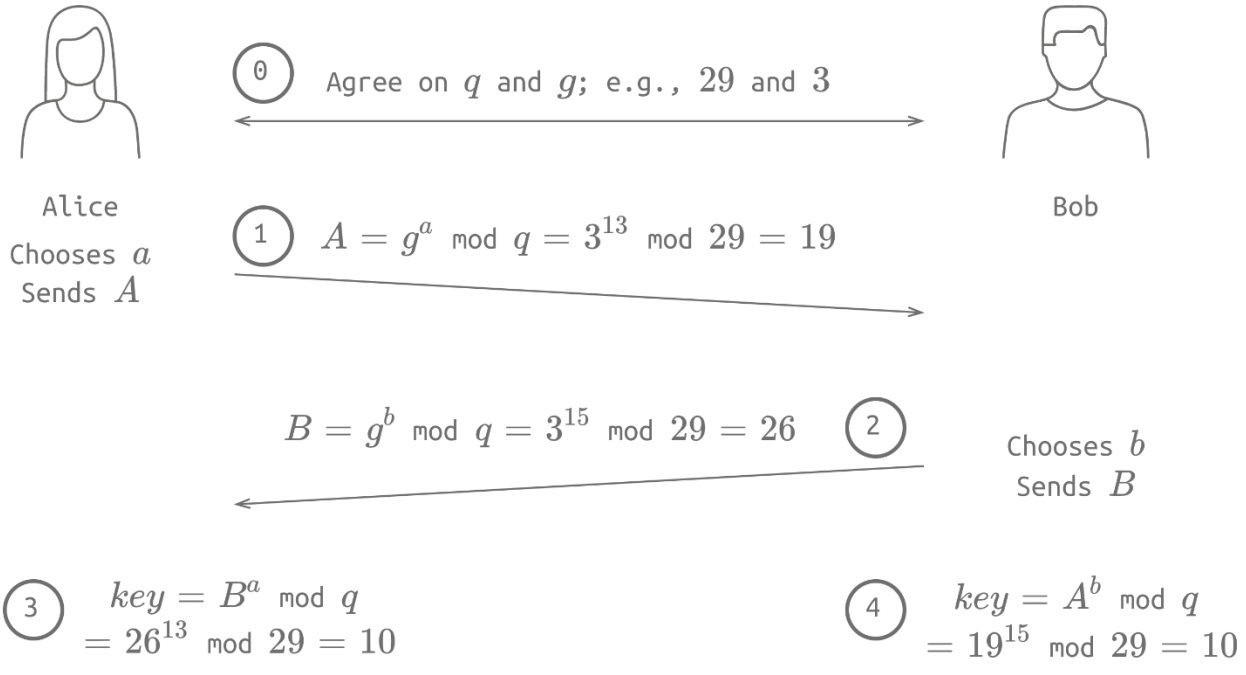
۳. باب یک عدد تصادفی b کمتر از q انتخاب می‌کند. او $B = (g^b) \bmod q$ را محاسبه می‌کند. باب باید b را مخفی نگه دارد؛ اما B را به آلیس ارسال می‌کند. فرض کنیم باب عدد $b = 15$ را انتخاب کرده و $B = 3^{15}\%29 = 26$ را محاسبه می‌کند. او B را به آلیس ارسال می‌کند.

۴. آلیس B را دریافت کرده و $key = B^a \bmod q$ را محاسبه می‌کند. مثال عددی $key = 26^{13} \bmod 29 = 10$.

۵. باب A را دریافت کرده و $key = A^b \bmod q$ را محاسبه می‌کند. مثال عددی $key = 19^{15} \bmod 29 = 10$.

ما می‌توانیم ببینیم که آلیس و باب به همان کلید دست یافتند.

اگرچه یک شنودگر مقادیر q، g، A و B را یاد گرفته است، اما قادر نخواهد بود کلید مخفی که آلیس و باب تبادل کرده‌اند را محاسبه کند. مراحل فوق در شکل زیر خلاصه شده است.



اگرچه اعدادی که انتخاب کرده‌ایم یافتن a و b را آسان می‌کند، حتی بدون استفاده از کامپیوتر، در نمونه‌های واقعی q با طول ۲۵۶ بیت انتخاب می‌شود. در اعداد اعشاری، این عدد ۱۱۵ با ۷۵ صفر در سمت راست آن است (نمی‌دانم چگونه بخوانم، اما به من گفته‌اند که به عنوان ۱۱۵ کواتورویگینتیلیون خوانده می‌شود). چنین q بزرگی یافتن a یا b را با وجود دانستن q، g، A و B غیرممکن می‌کند.

بیایید به پارامترهای واقعی دیفی-هلمن نگاهی بیندازیم. می‌توانیم از openssl برای تولید آن‌ها استفاده کنیم؛ باید گزینه dhparam را مشخص کنیم تا نشان دهیم که می‌خواهیم پارامترهای دیفی- هلمن را با اندازه مشخص شده در بیت، مانند ۲۰۴۸ یا ۴۰۹۶ تولید کنیم.

در خروجی کنسول زیر، می‌توانیم عدد اول P و تولیدکننده G را با استفاده از دستور openssl dhparam -in dhparams.pem -text -noout مشاهده کنیم. (این مشابه کاری است که با کلید خصوصی RSA انجام دادیم.)

```
Terminal

user@TryHackMe$ openssl dhparam -out dhparams.pem 2048
Generating DH parameters, 2048 bit long safe prime
[...]
$ openssl dhparam -in dhparams.pem -text -noout
    DH Parameters: (2048 bit)
    P:
        00:82:3b:9d:b5:29:31:f8:12:fe:21:e1:90:30:37:
        ac:d2:48:41:f7:d7:55:e5:d2:5d:dd:87:67:9e:bd:
        b3:97:df:05:a9:d2:d9:56:4f:66:b5:d9:d8:65:06:
        58:c3:8f:b3:0e:30:d2:9a:0b:c3:0a:56:8d:fc:0f:
        f2:e2:9e:4f:16:16:93:4e:b9:a4:c3:9c:09:2d:48:
        a2:ec:b6:97:92:63:a3:b4:75:36:3f:51:77:ca:ac:
        44:6d:99:eb:4d:4a:97:d5:4b:52:c8:07:f8:16:30:
        37:d3:b2:47:30:e6:4e:bc:6a:53:d1:9b:6a:4d:91:
        7a:4b:4f:af:3b:f0:ce:b9:ed:91:4d:8b:52:5a:3f:
        bb:6b:06:ae:32:95:7d:53:da:9b:ce:b0:ec:7d:81:
        25:05:d8:ce:ca:76:e7:d1:5a:31:13:d2:9f:62:b4:
        d5:ad:7d:cd:c9:ab:3d:28:e3:92:27:9f:f3:66:a0:
        be:61:49:cc:47:21:d8:e0:2c:e8:c6:35:4b:2f:ba:
        35:36:8f:bb:41:c6:89:b2:60:3c:62:bb:fe:bf:59:
        d3:7f:05:69:55:dc:61:1b:b4:bb:68:fa:65:1e:2e:
        46:2f:2d:21:62:d1:9f:a0:2b:aa:81:df:3a:f9:7d:
        0b:9d:0e:47:68:01:4f:6e:81:cc:4c:2a:91:fc:8c:
        f4:6f
    G:    2 (0x2)
```

الگوریتم مبادله کلید دیفی-هلمن به دو طرف اجازه می‌دهد تا بر سر یک راز از طریق یک کانال ناامن توافق کنند. با این حال، مبادله کلید بحث شده مستعد حمله مرد میانی (MitM) است؛ یک مهاجم ممکن است به آلیس پاسخ دهد و خود را به جای باب جا بزند و به باب نیز پاسخ دهد و خود را به جای آلیس جا بزند. ما راه‌حلی برای این مشکل را در کار ۶ بحث خواهیم کرد.

سوالات زیر را پاسخ دهید

به پوشه task 4 مراجعه نمایید.

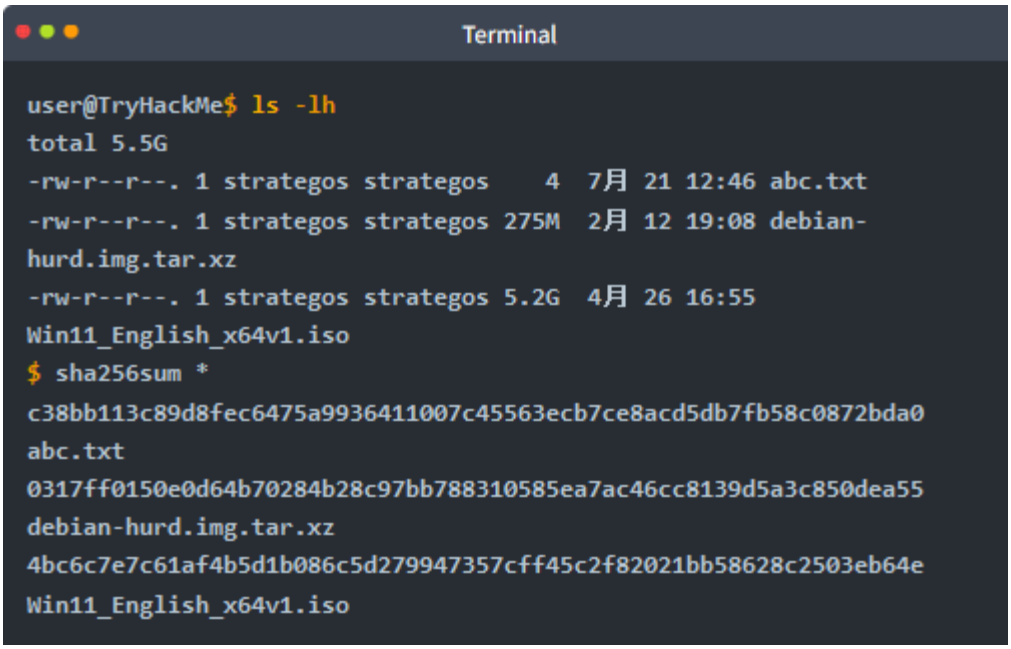
مجموعه‌ای از پارامترهای دیفی-هلمن در فایل dhparam.pem موجود است. اندازه عدد اول چند بیت است؟

ده بایت آخر عدد اول (LSB) چیست؟

بخش ۵: هشینگ

یک تابع هش رمزنگاری یک الگوریتم است که داده‌هایی با اندازه دلخواه را به عنوان ورودی می‌گیرد و یک مقدار با اندازه ثابت، به نام پیام خلاصه یا checksum، به عنوان خروجی برمی‌گرداند. به عنوان مثال، sha256sum خلاصه SHA256 (الگوریتم هش امن ۲۵۶) را محاسبه می‌کند. SHA256، همانطور که از نامش پیداست، یک checksum با اندازه ۲۵۶ بیت (۳۲ بایت) برمی‌گرداند. این checksum معمولاً با استفاده از ارقام هگزادسیمال نوشته می‌شود.

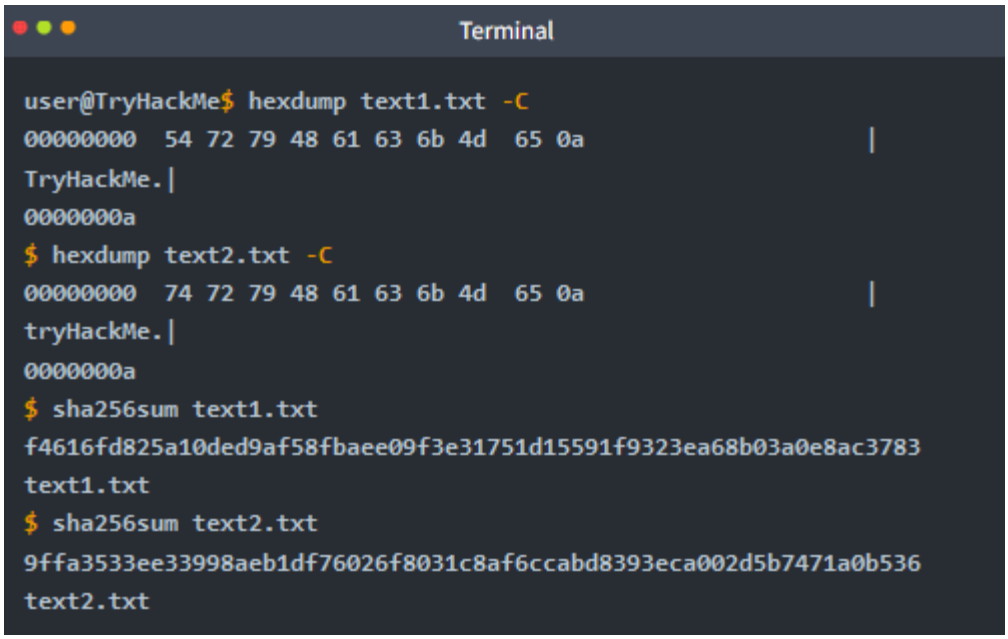
در خروجی ترمینال زیر، مقادیر هش SHA256 برای سه فایل با اندازه‌های متفاوت محاسبه شده است: ۴ بایت، ۲۷۵ مگابایت و ۵.۲ گیگابایت. با استفاده از sha256sum برای محاسبه پیام خلاصه هر سه فایل، سه مقدار کاملاً متفاوت به دست می‌آید که به نظر تصادفی می‌رسند. قابل توجه است که طول پیام خلاصه یا checksum به دست آمده یکسان است، صرف‌نظر از اینکه فایل چقدر کوچک یا بزرگ باشد. به‌ویژه، فایل چهار بایتی abc.txt و فایل ۵.۲ گیگابایتی، پیام‌های خلاصه با طول برابر تولید کردند که مستقل از اندازه فایل هستند.



هدف از چنین تابعی چیست؟ کاربردهای زیادی وجود دارد، به ویژه:

- ذخیره‌سازی پسوردها: به جای ذخیره‌سازی پسوردها به صورت متن ساده، هش پسورد ذخیره می‌شود. در نتیجه، اگر یک نقض داده اتفاق بیفتد، مهاجم لیستی از هش‌های پسورد به جای پسوردهای اصلی دریافت می‌کند.
- تشخیص تغییرات: هر تغییر جزئی در فایل اصلی باعث تغییر قابل توجهی در مقدار هش می‌شود.

در خروجی ترمینال زیر، دو فایل به نام‌های text1.txt و text2.txt داریم که تقریباً مشابه هستند به جز اینکه (به‌طور واقعی) یک بیت متفاوت دارند؛ حروف T و t در نمایش ASCII خود یک بیت را متفاوت نشان می‌دهند. حتی با تغییر فقط یک بیت، واضح است که checksum های SHA256 به‌طور کامل متفاوت هستند. بنابراین، اگر از یک الگوریتم تابع هش امن استفاده کنیم، می‌توانیم به‌راحتی تأیید کنیم که آیا تغییراتی رخ داده است یا خیر. این می‌تواند به محافظت در برابر دستکاری‌های عمدی و خطاهای انتقال فایل کمک کند.



چند الگوریتم هش که در حال حاضر مورد استفاده قرار می‌گیرند و هنوز هم امن به شمار می‌روند عبارتند از:

- SHA224، SHA256، SHA384، SHA512
- RIPEMD160

برخی از توابع هش قدیمی‌تر، مانند MD5 (Message Digest 5) و SHA-1، از لحاظ رمزنگاری شکسته شده‌اند. منظور از شکسته شدن این است که امکان تولید یک فایل متفاوت با همان checksum برای یک فایل خاص وجود دارد. این به این معنی است که می‌توانیم یک تصادم هش ایجاد کنیم. به عبارت دیگر، یک مهاجم می‌تواند یک پیام جدید با یک checksum معین ایجاد کند و تشخیص دستکاری فایل یا پیام ممکن نخواهد بود.

HMAC

کد تأیید هویت پیام مبتنی بر هش (HMAC) یک کد تأیید هویت پیام (MAC) است که علاوه بر تابع هش، از یک کلید رمزنگاری نیز استفاده می‌کند.

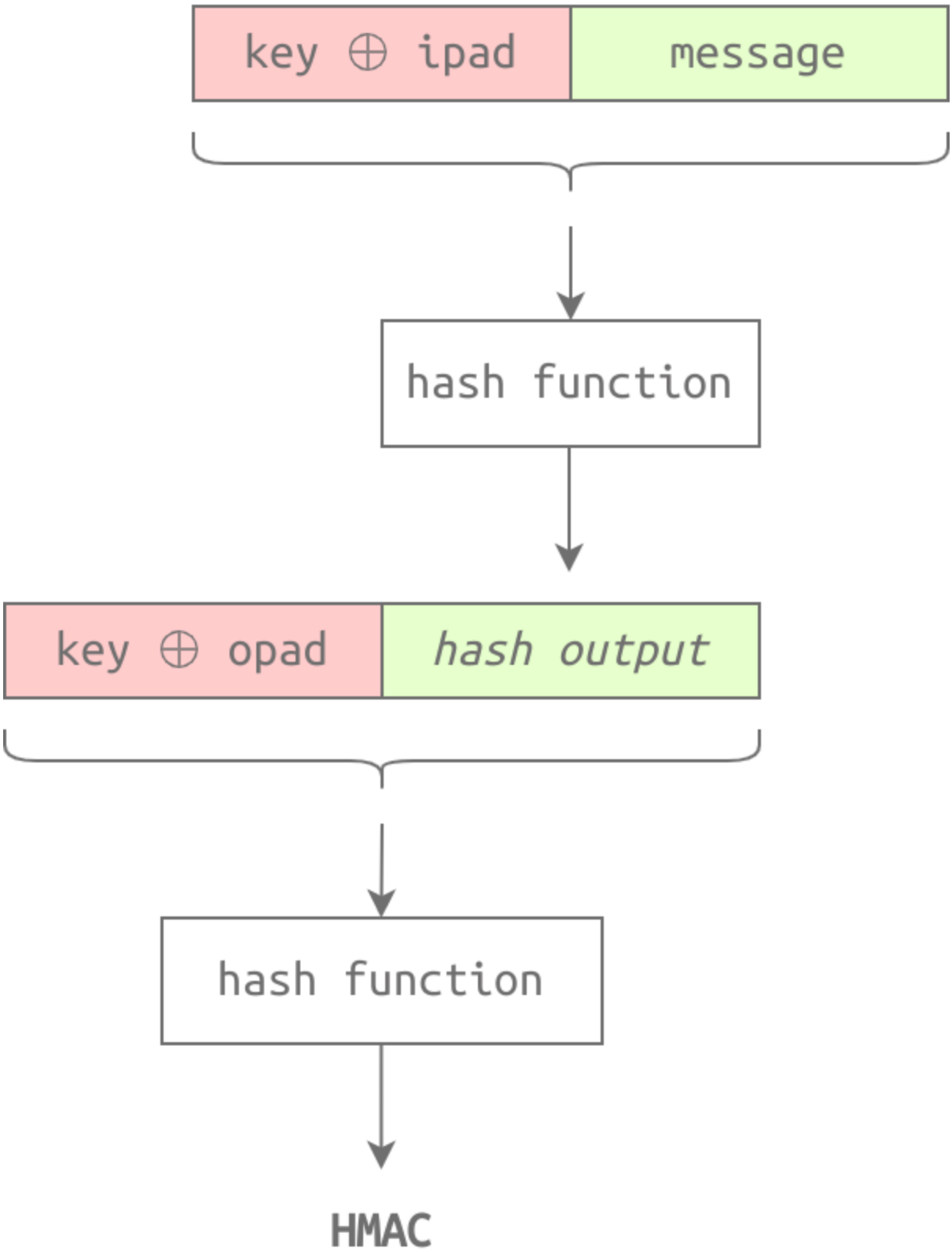
بر اساس RFC2104، HMAC به موارد زیر نیاز دارد:

- کلید مخفی
- پد داخلی (ipad) که یک رشته ثابت است. (RFC2104 از بایت x36۰ که به تعداد B تکرار شده است، استفاده می‌کند. مقدار B به تابع هش انتخاب شده بستگی دارد.)

- پد خارجی (opad) که یک رشته ثابت است. (RFC2104 از بایت x5C۰ که به تعداد B تکرار شده است، استفاده می‌کند.)

محاسبه HMAC مراحل زیر را دنبال می‌کند:

۱. اضافه کردن صفر به کلید تا طول آن برابر B شود، یعنی طول آن با طول ipad مطابقت داشته باشد.
۲. با استفاده از عملگر XOR، محاسبه کنید: key ⊕ ipad.
۳. پیام را به خروجی XOR مرحله ۲ اضافه کنید.
۴. تابع هش را به جریان بایت حاصل در مرحله ۳ اعمال کنید.
۵. با استفاده از XOR، محاسبه کنید: key ⊕ opad.
۶. خروجی تابع هش از مرحله ۴ را به خروجی XOR از مرحله ۵ اضافه کنید.
۷. تابع هش را به جریان بایت حاصل در مرحله ۶ اعمال کنید تا HMAC به دست آید.



شکل بالا نشان دهنده مراحل بیان شده در فرمول زیر است: H(K⊕opad,H(K⊕ipad,text)).

برای محاسبه HMAC در یک سیستم لینوکس، می‌توانید از هر یک از ابزارهای موجود مانند) hmac256 یا sha224hmac، sha256hmac، sha384hmac و (sha512hmac استفاده کنید، که در آن کلید مخفی بعد از گزینه key-- اضافه می‌شود.

به سوالات زیر پاسخ دهید

به پوشه task5 مراجعه کنید.

checksum SHA256 فایل order.json چیست؟

فایل order.json را باز کنید و مقدار ۱۰۰۰ را به ۹۰۰۰ تغییر دهید. checksum جدید SHA256 چیست؟

با استفاده از SHA256 و کلید RfDFz82۳، HMAC فایل order.txt چیست؟

بخش ۶: زیرساخت کلید عمومی (PKI) و SSL/TLS

استفاده از یک مبادله کلید مانند مبادله کلید دیفی-هلمن (Diffie-Hellman) به ما اجازه می‌دهد تا بر سر یک کلید مخفی توافق کنیم، در حالی که جاسوس‌ها می‌توانند این تبادل را زیر نظر داشته باشند. این کلید می‌تواند با یک الگوریتم رمزنگاری متقارن برای اطمینان از ارتباطات محرمانه استفاده شود. اما مبادله کلیدی که قبلاً توضیح دادیم در برابر حمله مرد میانه (MITM) مصون نیست. دلیل آن این است که آلیس (Alice) نمی‌تواند اطمینان حاصل کند که با باب (Bob) در حال ارتباط است و باب نیز نمی‌تواند اطمینان حاصل کند که با آلیس ارتباط برقرار کرده است.

به شکل زیر توجه کنید. این یک حمله علیه مبادله کلید توضیح داده شده در کار مبادله کلید دیفی-هلمن است. مراحل به این صورت است:

۱. آلیس و باب بر سر q و g توافق می‌کنند. هر کسی که در کانال ارتباطی گوش دهد می‌تواند این دو مقدار را بخواند، از جمله مهاجم، مالوری (Mallory).

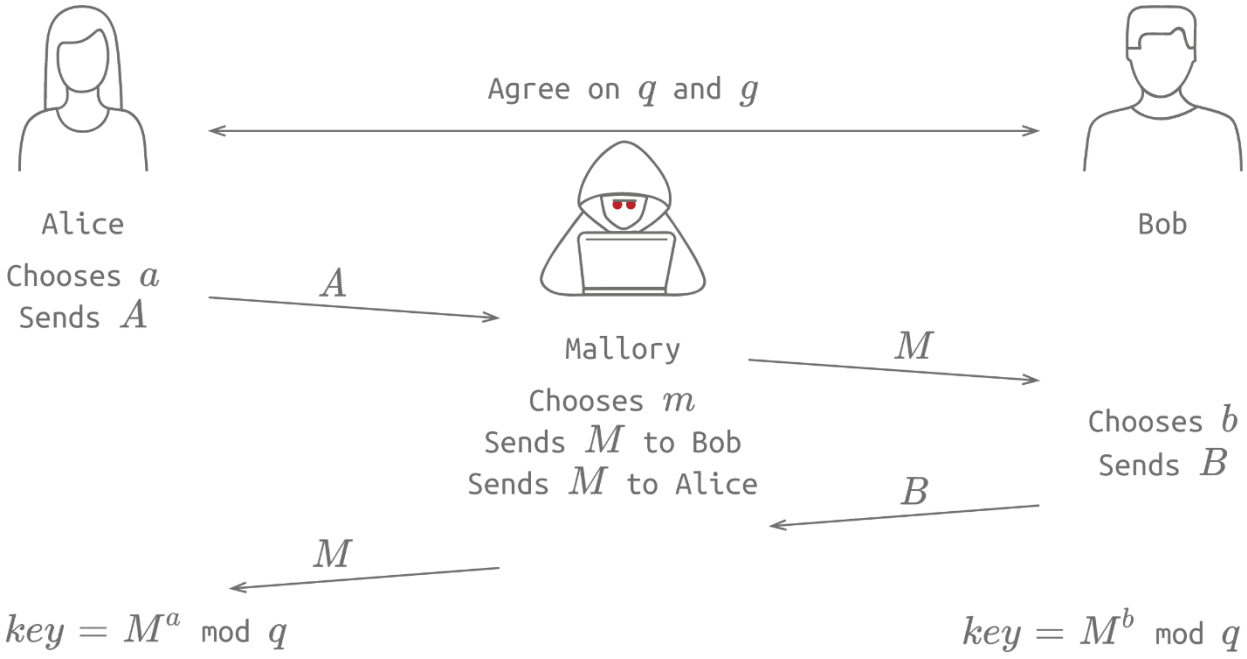
۲. همانطور که معمولاً انجام می‌دهد، آلیس یک متغیر تصادفی a را انتخاب می‌کند، A را محاسبه کرده (A = (g^a) mod q) و A را به باب ارسال می‌کند. مالوری منتظر این مرحله بوده و یک متغیر تصادفی m انتخاب کرده و M مربوطه را محاسبه می‌کند. به محض دریافت A، او M را به باب ارسال می‌کند و تظاهر می‌کند که آلیس است.

۳. باب M را دریافت کرده و فکر می‌کند که آلیس آن را ارسال کرده است. باب قبلاً یک متغیر تصادفی b انتخاب کرده و B مربوطه را محاسبه کرده است؛ او B را به آلیس ارسال می‌کند. به همین ترتیب، مالوری پیام را قطع کرده، B را می‌خواند و به جای آلیس M را به آلیس ارسال می‌کند.

۴. آلیس M را دریافت کرده و $key = M^a \bmod q$ را محاسبه می‌کند.

۵. باب M را دریافت کرده و $key = M^b \bmod q$ را محاسبه می‌کند.

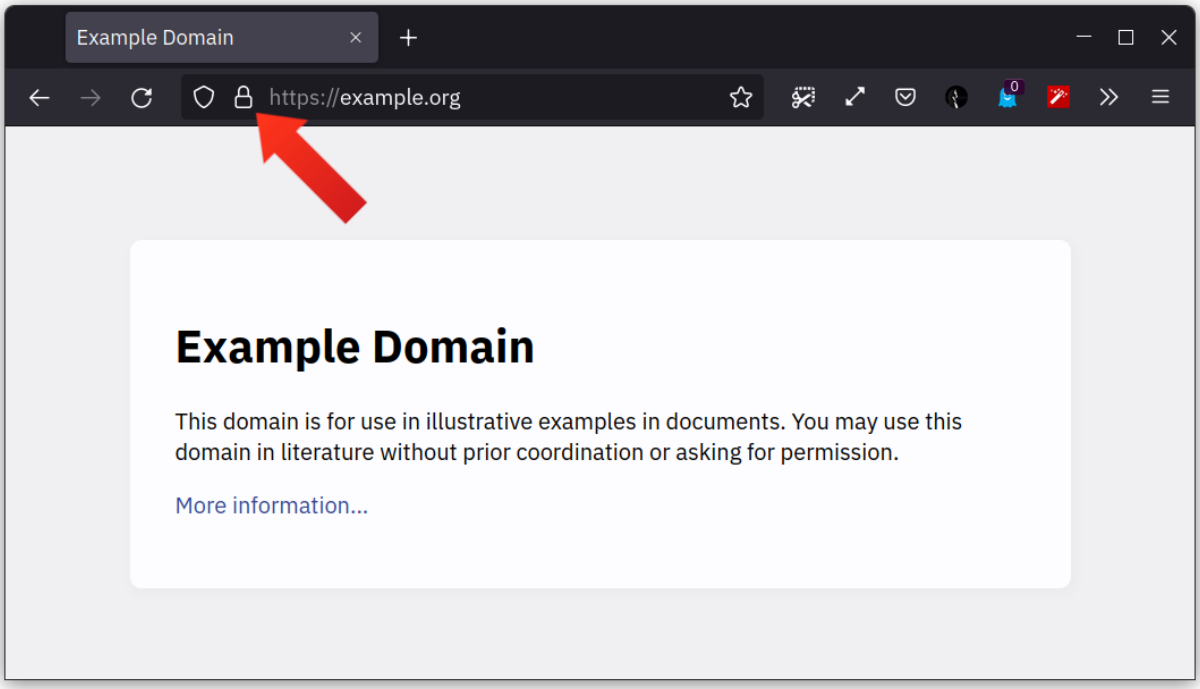
آلیس و باب به ارتباط خود ادامه می‌دهند و فکر می‌کنند که به طور مستقیم در حال ارتباط هستند، غافل از اینکه در حال ارتباط با مالوری هستند که می‌تواند پیام‌ها را بخواند و قبل از ارسال به گیرنده مورد نظر تغییر دهد.



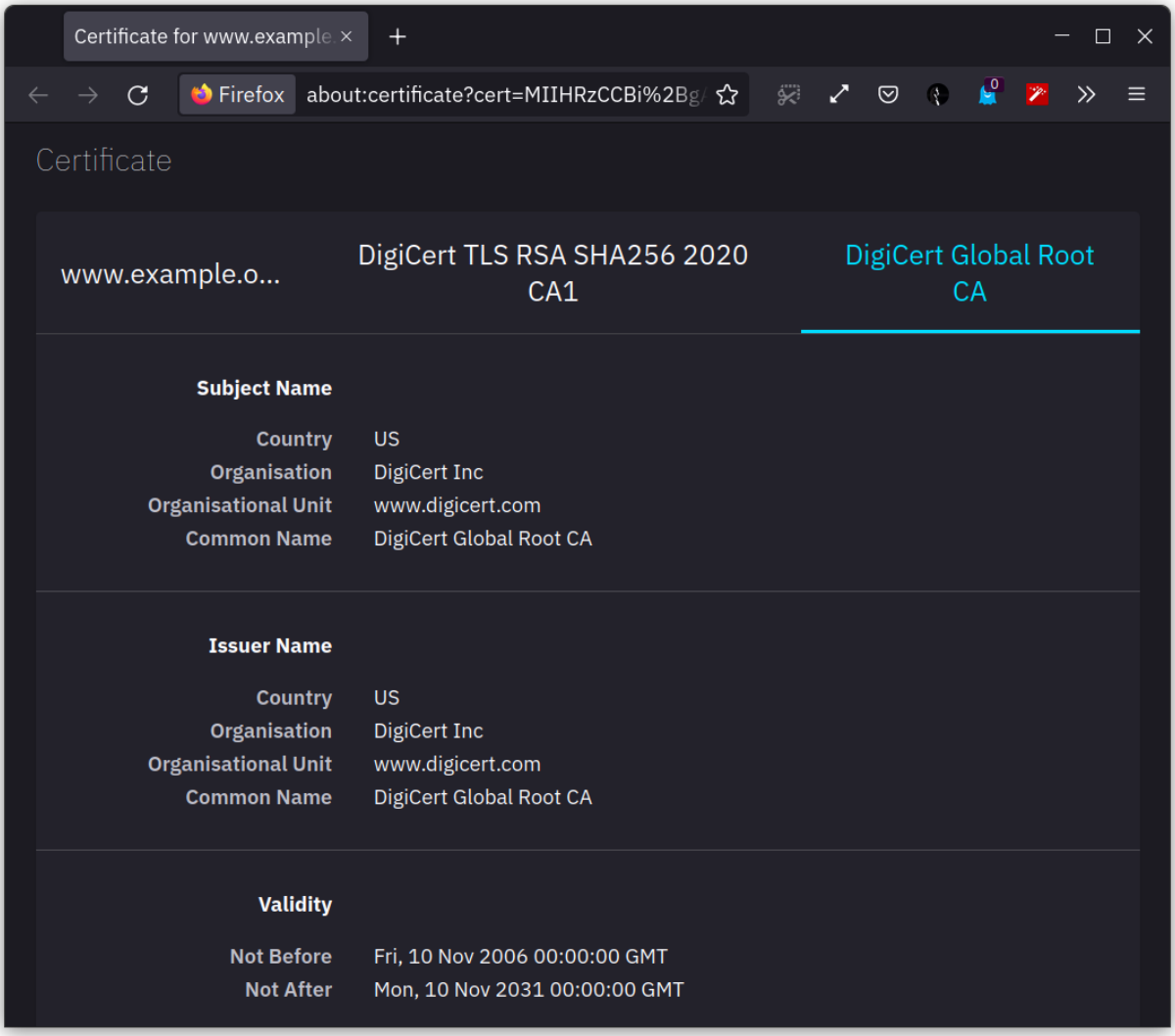
این آسیب‌پذیری نیاز به مکانیزمی را ایجاب می‌کند که به ما اجازه دهد هویت طرف مقابل را تأیید کنیم. این ما را به زیرساخت کلید عمومی (PKI) می‌کشاند.

به فرض اینکه شما در حال مرور وب‌سایت example.org از طریق HTTPS هستید. چگونه می‌توانید مطمئن باشید که در واقع با سرورهای example.org در حال ارتباط هستید؟ به عبارت دیگر، چگونه می‌توانید مطمئن باشید که هیچ مرد میانه‌ای بسته‌ها را قطع نکرده و قبل از رسیدن به شما تغییر نداده است؟ پاسخ در گواهی‌نامه وب‌سایت نهفته است.

تصویر زیر صفحه‌ای را نشان می‌دهد که هنگام مرور example.org به دست می‌آوریم. اکثر مرورگرها ارتباط رمزنگاری شده را با نوعی نماد قفل نشان می‌دهند. این نماد قفل نشان می‌دهد که ارتباط از طریق HTTPS با یک گواهی معتبر تأمین شده است.



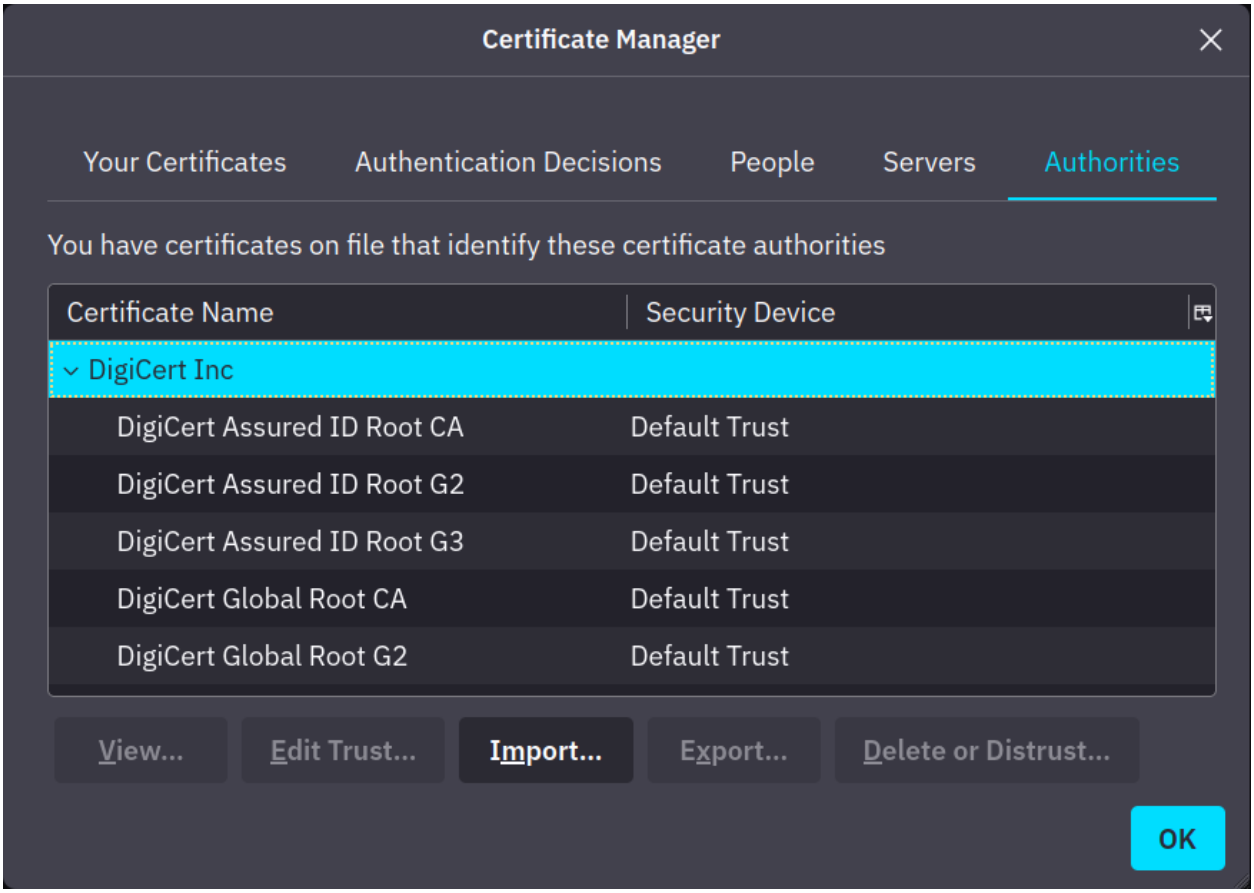
در زمان نگارش، وب‌سایت example.org از یک گواهی امضا شده توسط DigiCert Inc استفاده می‌کند. به عبارت دیگر، DigiCert تأیید می‌کند که این گواهی معتبر است (تا تاریخ مشخصی).



برای اینکه یک گواهی توسط یک مقام صدور گواهی (CA) امضا شود، ما نیاز داریم:

۱. درخواست صدور گواهی (CSR) را تولید کنیم: شما یک گواهی ایجاد کرده و کلید عمومی خود را برای امضا به یک طرف ثالث ارسال می‌کنید.
۲. CSR خود را به یک CA ارسال کنید: هدف این است که CA گواهی شما را امضا کند. راه‌حل جایگزین و معمولاً ناامن این است که گواهی خود را خودامضا کنید.

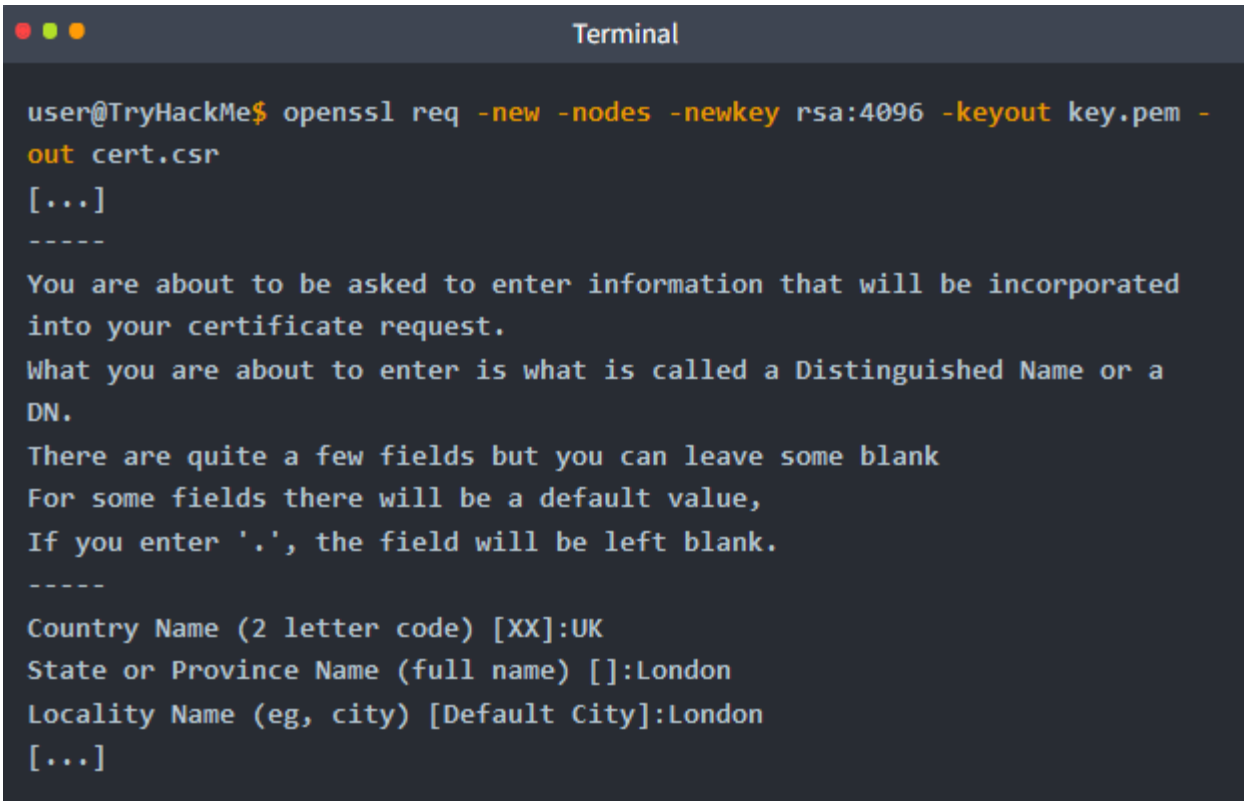
برای این کار، دریافت‌کننده باید CA که گواهی را امضا کرده است، شناسایی و اعتماد کند. و همانطور که انتظار می‌رود، مرورگر ما به DigiCert Inc به عنوان یک مقام امضا اعتماد دارد؛ در غیر این صورت، به جای ادامه به وب‌سایت درخواستی، یک هشدار امنیتی صادر می‌کرد.



شما می‌توانید از دستور openssl برای تولید یک درخواست امضای گواهی استفاده کنید openssl req -new : -nodes -newkey rsa:4096 -keyout key.pem -out cert.csr. از گزینه‌های زیر استفاده کرده‌ایم:

- req -new: ایجاد یک درخواست امضای گواهی جدید
- -nodes: ذخیره کلید خصوصی بدون گذرواژه
- -newkey: تولید یک کلید خصوصی جدید
- rsa:4096: تولید یک کلید RSA با اندازه ۴۰۹۶ بیت
- -keyout: مشخص کردن محل ذخیره کلید
- -out: ذخیره درخواست امضای گواهی

سپس از شما خواسته می‌شود تا به سوالاتی پاسخ دهید، همان‌طور که در خروجی کنسول نشان داده شده است.



پس از آماده شدن فایل CSR، می‌توانید آن را به CA دلخواه خود ارسال کنید تا امضا شده و آماده استفاده در سرور شما باشد.

پس از اینکه مشتری، یعنی مرورگر، یک گواهی امضا شده که به آن اعتماد دارد دریافت کرد، تبادل SSL/TLS انجام می‌شود. هدف این است که بر روی رمزها و کلید مخفی توافق کنیم.

ما به تازگی توضیح دادیم که PKI چگونه بر وب و گواهی‌های SSL/TLS اعمال می‌شود. یک طرف سوم مورد اعتماد برای مقیاس‌پذیری سیستم ضروری است.

برای اهداف آزمایشی، ما یک گواهی خودامضا ایجاد کرده‌ایم. به عنوان مثال، دستور زیر یک گواهی خودامضا تولید می‌کند:

openssl req -x509 -newkey -nodes rsa:4096 -keyout key.pem -out cert.pem -sha256 -days 365

x509- نشان می‌دهد که می‌خواهیم یک گواهی خودامضا به جای درخواست گواهی تولید کنیم.

-sha256 استفاده از digest SHA-256 را مشخص می‌کند. این گواهی برای یک سال معتبر خواهد بود زیرا -days 365 را اضافه کرده‌ایم.

برای پاسخ به سوالات زیر، شما باید فایل گواهی cert.pem را در دایرکتوری task06 بررسی کنید. می‌توانید از دستور زیر برای مشاهده گواهی خود استفاده کنید:

```
openssl x509 -in cert.pem -text
```

به سوالات زیر پاسخ دهید (به پوشه task 6 مراجعه شود.)

اندازه کلید عمومی به بیت چقدر است؟

این گواهی تا کدام سال معتبر است؟

بخش ۷: احراز هویت با استفاده از گذرواژه‌ها

بیایید ببینیم چگونه رمزنگاری می‌تواند به افزایش امنیت گذرواژه‌ها کمک کند. با PKI و SSL/TLS، می‌توانیم با هر سرور ارتباط برقرار کنیم و اعتبارنامه‌های ورود خود را ارائه دهیم، در حالی که اطمینان داریم هیچ‌کس نمی‌تواند گذرواژه‌های ما را در حین انتقال در شبکه بخواند. این یک مثال از حفاظت از داده‌ها در حین انتقال است. بیایید بررسی کنیم چگونه می‌توانیم گذرواژه‌ها را در حین ذخیره‌سازی در یک پایگاه داده ایمن نگه داریم، یعنی داده‌ها در حالت استراحت.

ناامن‌ترین روش این است که نام کاربری و گذرواژه را در یک پایگاه داده ذخیره کنیم. به این ترتیب، هرگونه نفوذ به داده‌ها، گذرواژه‌های کاربران را افشا می‌کند. هیچ تلاشی فراتر از خواندن پایگاه داده حاوی گذرواژه‌ها مورد نیاز نیست.

گذرواژه	نام کاربری
qwerty	alice
dragon	bob
princess	charlie

روش بهبود یافته این است که نام کاربری و نسخه هش‌شده گذرواژه را در یک پایگاه داده ذخیره کنیم. به این ترتیب، در صورت نفوذ به داده‌ها، نسخه‌های هش‌شده گذرواژه‌ها افشا خواهند شد. از آنجایی که تابع هش غیرقابل بازگشت است، مهاجم باید تلاش کند گذرواژه‌های مختلفی را امتحان کند تا گذرواژه‌ای را پیدا کند که همان هش را تولید کند. جدول زیر MD5 هش گذرواژه‌ها را نشان می‌دهد. (ما MD5 را انتخاب کردیم تا این مثال اندازه فیلد گذرواژه کوچک باشد؛ در غیر این صورت، ما از SHA256 یا چیزی امن‌تر استفاده می‌کردیم.)

نام کاربری	هش(گذرواژه)
alice	d8578edf8458ce06fbc5bb76a58c5ca4
bob	8621ffdbc5698829397d97767ac13db3
charlie	8afa847f50a716e64932d995c8e7435

روش قبلی به نظر ایمن می‌آید؛ با این حال، در دسترس بودن جدول‌های رنگین (rainbow tables) این روش را ناامن کرده است. یک جدول رنگین شامل لیستی از گذرواژه‌ها همراه با مقدار هش آن‌ها است. بنابراین، مهاجم تنها نیاز دارد که هش را جستجو کند تا گذرواژه را بازیابی کند. به عنوان مثال، جستجوی d8578edf8458ce06fbc5bb76a58c5ca4 برای کشف گذرواژه اصلی alice بسیار آسان خواهد بود. بنابراین، ما نیاز داریم روش‌های ایمن‌تری برای ذخیره‌سازی گذرواژه‌ها پیدا کنیم؛ می‌توانیم مقدار تصادفی الحاقی (salt) اضافه کنیم. مقدار تصادفی الحاقی یک مقدار تصادفی است که می‌توانیم آن را قبل از هش کردن به گذرواژه اضافه کنیم. مثالی در زیر نشان داده شده است.

نام کاربری	هش(گذرواژه + مقدار تصادفی الحاقی)	مقدار تصادفی الحاقی
alice	8a43db01d06107fcad32f0bcfa651f2f	12742
bob	aab2b680e6a1cb43c79180b3d1a38beb	22861
charlie	3a40d108a068cdc8e7951b82d312129b	16056

جدول بالا از hash(گذرواژه + مقدار تصادفی الحاقی) استفاده کرده است؛ روش دیگری که می‌توان استفاده کرد، hash(hash(گذرواژه) + مقدار تصادفی الحاقی) است. توجه داشته باشید که ما از مقدار تصادفی الحاقی نسبتاً کوچکی همراه با تابع هش MD5 استفاده کردیم. اگر این یک راه‌اندازی واقعی بود، باید به یک تابع هش (بیشتر) امن و مقدار تصادفی الحاقی بزرگ‌تر برای امنیت بهتر تغییر می‌کردیم.

بهبود دیگری که می‌توانیم قبل از ذخیره‌سازی گذرواژه انجام دهیم، استفاده از تابع مشتق‌سازی کلید (KDF) مانند PBKDF2 (Password-Based Key Derivation Function 2) است. PBKDF2 گذرواژه و مقدار تصادفی الحاقی را می‌گیرد و آن را از طریق تعدادی تکرار، معمولاً صدها هزار بار، عبور می‌دهد.

توصیه می‌کنیم اگر می‌خواهید در مورد تکنیک‌های دیگر مربوط به ذخیره‌سازی گذرواژه یاد بگیرید، به "Password Storage Cheat Sheet" مراجعه کنید.

به سوالات زیر پاسخ دهید

شما در حال بررسی یک سیستم بودید که متوجه شدید هش MD5 گذرواژه مدیر 3fc0a7acf087f549ac2b266baf94b8b1 است. گذرواژه اصلی چیست؟

بخش ۸: رمزنگاری و داده – مثال

در این بخش، می‌خواهیم بررسی کنیم چه اتفاقی می‌افتد وقتی که به یک وب‌سایت از طریق HTTPS وارد می‌شویم.

۱. کلاینت درخواست گواهی SSL/TLS سرور را می‌دهد.
۲. سرور گواهی SSL/TLS را به کلاینت ارسال می‌کند.
۳. کلاینت تأیید می‌کند که گواهی معتبر است.

نقش رمزنگاری از بررسی گواهی آغاز می‌شود. برای اینکه یک گواهی معتبر تلقی شود، باید امضا شده باشد. امضا کردن به معنای این است که یک هش از گواهی با کلید خصوصی یک طرف سوم معتبر رمزگذاری شده و هش رمزگذاری شده به گواهی ضمیمه می‌شود.

اگر طرف سوم معتبر باشد، کلاینت از کلید عمومی آن برای رمزگشایی هش رمزگذاری شده استفاده می‌کند و آن را با هش گواهی مقایسه می‌کند. با این حال، اگر طرف سوم شناسایی نشود، اتصال به طور خودکار ادامه نخواهد یافت.

پس از اینکه کلاینت تأیید کرد که گواهی معتبر است، یک دست دادن SSL/TLS آغاز می‌شود. این دست دادن به کلاینت و سرور اجازه می‌دهد تا بر روی کلید مخفی و الگوریتم رمزنگاری متقارن توافق کنند، به علاوه موارد دیگر. از این نقطه به بعد، تمامی ارتباطات مربوط به جلسه با استفاده از رمزنگاری متقارن رمزگذاری می‌شود.

گام نهایی ارائه اعتبارنامه‌های ورود است. کلاینت از جلسه رمزگذاری شده SSL/TLS برای ارسال آن‌ها به سرور استفاده می‌کند. سرور نام کاربری و گذرواژه را دریافت کرده و نیاز دارد تأیید کند که آن‌ها با هم مطابقت دارند.

با رعایت دستورالعمل‌های امنیتی، انتظار داریم سرور نسخه هش‌شده گذرواژه را پس از اضافه کردن یک مقدار تصادفی الحاقی تصادفی ذخیره کند. به این ترتیب، اگر پایگاه داده به خطر بیفتد، بازیابی گذرواژه‌ها چالش‌برانگیز خواهد بود.

موفق باشید.